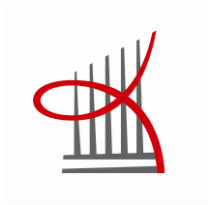


Laadunvarmistus sähköverkkoyhtiön tietojärjestelmäprojektissa

Lauri Anttila

Opinnäytetyö
Joulukuu 2011
Tietojärjestelmäosaamisen koulutusohjelma
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojärjestelmäosaamisen YAMK-koulutusohjelma

ANTTILA, LAURI: Laadunvarmistus sähköverkkoyhtiön tietojärjestelmäprojektissa

Opinnäytetyö 81 s
Joulukuu 2011

Tämän opinnäytetyön taustalla on Vattenfall Verkko Oy:n tarve korvata sähköverkon rakentamiseen ja kunnossapitoon käytetty Avux-työnohjausjärjestelmä uudella ja tehostaa sähköistä sanomaliikennettä sopimusurakoitsijoiden kanssa. Uudistuksen yhteydessä otetaan käyttöön vuonna 2010 laadittu XML-sanomastandardi sähköverkkoyhtiöiden ja niiden aliurakoitsijoiden väliseen työtilausprosessiin. Järjestelmäuudistukseen liittyen opinnäytetyö pureutuu uuden urakoitsija-alusta Wormsin laadunvarmistukseen sen ohjelmointi- ja käyttöönottovaiheessa.

Testauksen lähtökohtana oli järjestelmän toiminnan kokonaisvaltainen analysointi, joka huomioi myös tulevaisuuden kasvavat vaatimukset käyttäjä- ja tiedonsiirtokapasiteetin osalta. Ennen opinnäytetyön aloittamista tehdyn vaatimusmäärittelyn tuloksena syntynyt vaatimuslista käytiin läpi testaajien toimesta testauksen eri vaiheissa. Järjestelmässä on monen tyyppisiä integraatioita eri järjestelmiin (mm. SAP) sekä paljon järjestelmien välillä liikkuvien sanomien tulkintaa.

Testaus nojaa FURPS-mallin pääperiaatteisiin ja käy läpi järjestelmän eri osa-alueet kohta kohdalta. Kun järjestelmän eri osat oli testattu erikseen, tehtiin koko järjestelmälle vielä kattava sarja erilaisia ns. end-to-end -testejä, joilla pyrittiin simuloimaan todellista käyttöä.

Sekä testattava järjestelmä että testityökalut ovat pääosin avoimeen lähdekoodiin perustuvia ja siten mielenkiintoisia vaihtoehtoja kalliiden kaupallisten ratkaisujen rinnalle. Ne asettavat järjestelmälle myös haasteita, koska tukipalvelut ovat vähäisiä ja vaativat organisaatiolta omaa osaamista.

Opinnäytetyö osoittaa, että hieman laajempikin ohjelmistoprojekti pystytään testaamaan tilaajaorganisaation toimesta ilman ulkopuolelta ostettua kallista testausapua. Tämä vaatii tilaajalta ainoastaan asiaan vihkiytyneen resurssin, jolla on riittävä tekninen tuntemus tietojärjestelmien toiminnasta.

Vaikka järjestelmä on käyttöönottovaiheessa vaatimusten mukainen, ei testaustyötä sovi unohtaa tulevaisuudessakaan. Erilaisten lisäominaisuuksien ja muiden päivitysten myötä on tärkeää, että toiminnallisuus ja suorituskyky säilyvät myös jatkossa sellaisina, kuin ne on alun perin suunniteltu olevan.

Asiasanat: ohjelmistotestaus, FURPS, kuormatestaus, tietojärjestelmät, laadunhallinta

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Master's Degree in Information Systems Competence

ANTTILA, LAURI: Quality Management in Software Project for Electricity Distribution Company

Master's Thesis 81 pages
December 2011

The purpose of the thesis work was to monitor the quality of a new software implementation – Worms – for Vattenfall Verkko Oy. It is a system designed to send and receive work order related information to and from external contractors. The new system will replace an old system called Avux and it relies on an XML standard created during 2010 in cooperation with grid companies and their contractors.

To produce as good a system as possible, theoretical base for use of FURPS quality handling model was studied and then implemented in a series of tests. Future users of the system were involved to measure usability and find defects during the coding phase, so that everything would function immediately, when the system was taken into action.

Both the system and testing tools are mostly based on open source technologies, making them an interesting alternative to expensive commercial applications. During the testing, this kind of approach was found to be very good. However, using open source adds more pressure on the IT maintenance people, since it does not have the kind of support as commercial versions have. This should be taken into account when selecting a new software platform.

One key finding is that companies do not need external help to test their software. They just need someone within their organization who has knowledge about technical issues related to software and time to learn some basics of testing.

As a result of this thesis, Vattenfall Verkko has new software which functions as described in original functional requirements and which is very stable and reliable. This guarantees better service level from external contractors, since they have more real-time data of their work orders, which is also available when they need it.

Even the project itself is concluded, the quality control and improvements should definitely not stop here. To achieve best value for the investment, it is feasible to keep on testing the system and prepare for future upgrades in advance.

Key words: FURPS, software quality, load testing, information systems, quality control

SISÄLTÖ

KÄSITTEET

AJAX
Ajaxista puhuttaessa viitataan ohjelmointitekniikkojen kokonaisuuteen, joka koostuu HTML:stä, CSS:stä, DOMista, XML:stä, XMlHttpRequestista ja JavaScriptistä. Sitä käytetään tiedon dynaamiseen esittämiseen ja muuttamiseen web-ohjelmistoissa.

Apache Camel
Camel on avoimen lähdekoodin koodikirjasto (framework), joka keskittyy integraatiorajapintojen (etenkin SOAP/XML) tehokkaaseen hallintaan.

Apache Tomcat
Avoimen lähdekoodin palvelinohjelmisto, joka mahdollistaa Java Servletien ajamisen palvelimella.

Apache Wicket
Framework mm. Java-pohjaisten ohjelmistojen web-käyttöliittymän kehitykseen.

DMS
Tekla NIS -järjestelmässä käytössä oleva käytöntukisovellus, joka havaitsee vikoja sähköverkossa ja jolla käyttökeskus voi tilata viankorjaustöitä palveluntuottajayhtiöiltä.

DOM
Document Object Model, tapa esittää ja manipuloida tietoa dynaamisesti (kts. Ajax)

ebMS
ebXML Message Service määrittelee kuoren (envelope) ja otsikot (header) ebXML-viestien siirtämiseksi osapuolten välillä. Se rakentuu SOAP- ja SOAPAttach-protokollien päälle. ebMS on myös riippumaton tiedonsiirtotavasta (esim. HTTP, FTP, SMTP)

ebXML
Lyhenne muodostuu sanoista Electronic Business using eXtensible Markup Language. Se on Oasis-yhteisön kehittämä avoin XML-standardi, joka on tarkoitettu kaikenlaisen sähköisen liiketoiminnan harjoittamiseen.

FTP
File Transfer Protocol on yksi yleisimpiä tiedostojen siirtoon tarkoitettuja tekniikoita.

HTTP
Hypertext transfer protocol on www-sivujen esitystapa. Salattu versio on nimeltään HTTPS.

Käyttökeskus
Sähköverkon toimintaa kokonaisuudessaan valvova verkkoyhtiön toiminto.

NIS
Network Information System, eli verkkotietojärjestelmä, jonka avulla hallinnoidaan sähkö- tai tietoverkon verkostokomponentteja.

Java
Java on yksi maailman yleisimmin käytettyjä ohjelmointikieliä. Sen ensimmäisen version on julkaissut Sun Microsystems vuonna 1995. Kieltä käytetään monipuolisesti erilaisilla aluistoilla tietokoneista puhelimiin.

Jenkins
Versiohallintatyökalu ohjelmoijan avuksi.

Jmeter
Apache Foundationin avoimen lähdekoodin testiohjelmisto.

MySQL
MySQL on avoimen lähdekoodin SQL-tietokanta. Se on yksi maailman yleisimmin käytetyistä tietokantaratkaisuista.

OVT-tunnus
Organisaatioiden Välinen Tiedonsiirtotunnus. Käytetään yleisimmin sähköisissä laskuissa, mutta esimerkiksi Vattenfallin tapauksessa sen avulla ohjataan työtilaukset oikealle urakoitsijalle. Koostuu Suomen verohallinnon ISO 6523 mukaisesta koodista (0037) ja yrityksen Y-tunnuksesta ilman FI-etuliitettä ja väliviivaa.

RPC
Remote Procedure Call on järjestelmien välinen kommunikointitapa, jossa client-järjestelmä aloittaa tiedonvaihdon serverin kanssa, eli toimii aktiivisena osapuolena.

SALT
Salasanojen yksilöimiseen käytetty salaustapa, joka lisää käyttäjän määrittämän salasanan jatkeeksi sattumanvaraisia merkkejä. Tämä parantaa käyttäjätietokannan tietoturvaa, koska erilaiset siihen kohdistuvat hyökkäykset muuttuvat vaikeammiksi toteuttaa.

Schema
Schemalla määritetään XML-viestin rakenne asettamalla eri elementeille tyyppejä ja pakollisuussääntöjä. Yksi XML-viesti voi hyödyntää useampaa schemaa.

Scrum
Ketterään (agile) ohjelmistokehitykseen kehitetty menetelmä, jolla pyritään tuottamaan nopeasti uusia testiversioita ohjelmistosta asiakkaan testattavaksi ja tämän jälkeen jatkojalostaa tuotteesta aina vain valmiimpia versioita.

SOAP
Sanoista Simple Object Access Protocol muodostuva SOAP on protokolla, jolla tietojärjestelmät voivat vaihtaa keskenään informaatiota. Sen toiminta perustuu XML-tiedostoihin. Sen sijaan, että XML-tiedostot tallennettaisiin esimerkiksi palvelimelle

tiettyyn kansioon, josta järjestelmä ne lukisi, SOAPin avulla data voidaan tallentaa suoraan vastaanottajan tietojärjestelmään ilman erillistä käsittelyä. Tämä mahdollistaa myös reaaliaikaisen kommunikoinnin järjestelmien välillä XML-muodossa.

SoapUI
Avoimen lähdekoodin testiohjelmisto, jolla voidaan simuloida SOAP-liikennettä järjestelmien välillä.

Sonar
Ohjelmistokehitystyökalu, jolla voidaan valvoa koodin rakennetta.

UBL
Unified Business Language on XML-kirjasto, joka sisältää suuren määrän yritystenvälisessä tietoliikenteessä tarvittavia tietueita. Sähköverkkoyhtiöiden TIEKE XML-standardi perustuu UBL 2.0-kirjaston osista koottuun kokonaisuuteen.

Urakoitsija
Sähköverkkoyhtiön sopimuskumppani, joka vastaa käytännössä sähköverkon rakentamisesta ja korjaamisesta.

Validointi
Koodin tai sanoman tarkastus virheiden varalta.

Välityspalvelu / broker
Kolmannen osapuolen tarjoama palvelu, joka välittää esim. XML-sanomat osapuolelta toiselle, poistaen tarpeen luoda suuria määriä point-to-point -yhteyksiä.

XML
Sanoista eXtensible Markup Language muodostuva XML on yleisnimitys selkokielisistä sanomista, joita vaihdetaan tietojärjestelmien välillä. Sitä käytetään etenkin toisistaan eroavien järjestelmien välisessä tietoliikenteessä ”yhteisenä kielenä”. Käytännössä XML:n sisältö voi olla mitä hyvänsä, mutta eri aloille on laadittu omia standardeja.

XMLHttpRequest
Tiedonsiirtotapa, jolla dataa siirretään selaimen ja palvelimen välillä.

1 JOHDANTO

Sähköverkon rakentaminen ja kunnossapito ovat nyky-yhteiskunnassa elintärkeitä asioita. On vaikea kuvitella elämää ilman sähköä, mutta sääilmiöistä ja muista syistä johtuen ei sähkönjakelua voida sataprosenttisesti taata jatkuvasti kaikille. Luotettavan sähkönjakelun perustana ovat toimivat järjestelmät ja sähköverkon kunnosta vastaavat tahot.

Vattenfall Verkko Oy:n hallinnoima sähköverkko ja toimintaympäristö ovat kehittyneet vuosien mittaan todella vauhdikkaasti. Tietojärjestelmien kehitykselle tämä on asettanut haasteita ja etenkin työnohjaukseen liittyvien järjestelmien elinkaari on tullut tiensä päähän. Alun perin väliaikaiseksi ratkaisuksi tarkoitettu Avux-työnhallintajärjestelmä ei enää pysy mukana jatkuvasti kehittyvien liiketoimintatarpeiden, tietojärjestelmäintegraatioiden ja datamäärän vauhdissa. Tästä syystä syntyi tämäkin opinnäytetyö, joka pureutuu Avuxin korvaamisen haasteisiin nykyisessä toimintaympäristössä, sekä ohjelmistoratkaisuihin joiden toiminta saadaan optimoitua vastaamaan jatkuvasti kehittyviin ympäristön ja lainsäädännön vaatimuksiin.

Opinnäytetyö keskittyy sähköverkkoyhtiön ja sen monien sopimuskumppanien väliseen kommunikointiin tarkoitetun ”Worms” (Work Order Management System) -järjestelmän käyttöönottovaihetta edeltävään laadunvarmistukseen. Taustalla on käynnissä mittavia uudistuksia koko työnohjausprosessiin, jotka toteutuessaan tehostavat toimintaa merkittävästi. Lopulta asiakkaille voidaan tarjota entistä luotettavampaa sähköverkkopalvelua, eli lyhyempiä sähkökatkoja. Lisäksi eri palvelutuottajien (myöhemmin urakoitsijoiden) vertailu keskenään tehostuu ja mahdollistaa tehokkaamman kilpailuttamisen Vattenfall Verkko Oy:n toiminta-alueella.

Opinnäytetyön lopputuloksena on vaatimusten mukaisesti toimiva, laatuvaatimukset täyttävä tietojärjestelmä, joka skaalautuu tarvittaessa käsittelemään helposti nykyistä suurempia määriä liikennettä ja mukautuu vaivattomasti toimintaympäristössä tapahtuviin muutoksiin.

2 VATTENFALL VERKKO OY

Työn toimeksiantaja Vattenfall Verkko Oy on Suomen suurimpia sähköverkonhaltijoita. Sen kotipaikka on Tampereella, jonka lisäksi pieniä toimipisteitä on Hämeenlinnassa, Jyväskylässä, Oulaisissa, Heinolassa, Seinäjoella ja Viitasaarella. Yritys on osa ruotsalaista Vattenfall-konsernia.

2.1 Historia

Vuonna 1995 Suomessa astui voimaan uusi sähkömarkkinalaki, joka avasi sähkömarkkinat vapaalle kilpailulle. Vaiheittain käyttöönotettu laki mahdollisti jokaisen kuluttajan valitsemaan itse haluamansa sähkönmyyjän, kun aiemmin kuluttaja oli ollut sidoksissa oman alueensa sähköverkonhaltijan tuottamaan sähköön (Energiamarkkinavirasto 2010).

Muuttunut lainsäädäntö mahdollisti Vattenfallin rantautumisen Ruotsista Suomeen sähkömarkkinoiden perässä. Aiemmin sähköä oli myyty Suomeen jo pohjoismaisen Nordel-yhteistyön puitteissa, mutta lähtiessään omana itsenään mukaan sähkön myyntitoimintaan Suomessa halusi yhtiö myös osansa sähkön jakeluverkoista. Vuonna 1995 Vattenfall osti kuntien omistuksissa olleet Lapuan Sähkön sekä Hämeen Sähkön. Vuonna 1999 yhtiöön sulautettiin Heinolan Energia ja Revon sähkö. Lopulta vuonna 2000 hankittiin vielä Keski-Suomen Valo sekä Hämeenlinnan Energia. Sähkömarkkinalain vaatimuksesta verkkoyhtiöt muodostivat Vattenfall Verkko Oy:n, kun sähköenergian myynti keskittyi Vattenfall Sähkönmyynti Oy:lle. (Vattenfall 2010)

Nykyinen Vattenfall Verkko koostuu siis melko laajasta kirjosta eri yrityksiä, joten yrityksen on täytynyt luoda yhtenäinen yrityskulttuuri sekä yhtenäistää prosessit ja tietojärjestelmät tukemaan nykymuotoista toimintaa.

2.2 Tunnuslukuja

Henkilöstö	170hlö (kumppanit n. 600-700hlö, erityistilanteissa 1000hlö)
Liikevaihto (2010)	214milj. euroa
Asiakkaita	395 000
Markkinaosuus	12 % sähköverkon käyttäjistä Suomessa
Toiminta-alue	n. 50 000 neliökilometriä / 100 kuntaa
Sähköverkkoa	n. 61 000 km (155 metriä / asiakas)
Sähköasemia	135 kpl
Muuntamoita	21 317 kpl

2.3 Toimintaympäristö

Vattenfall Verkko vastaa sähkönjakelusta omalla toimialueellaan, joka kattaa leveän kaistan Etelä-Hämeestä Pohjois-Pohjanmaalle. Yritys toimii tarkasti säännellyllä alalla, jossa toimintaa rajoittaa sähkömarkkinalaki. Yritys ei monopoliasemansa vuoksi saa esimerkiksi tuottaa liikaa voittoa, vaan Energiamarkkinavirasto on määritellyt sallitun tuoton.

Uuden sähköverkon rakentamisesta ja kunnossapidosta huolehtivat alihankintana useat urakoitsijayritykset, joista suurimpia sekä tunnetuimpia ovat Empower Oy ja Eltel Networks Oy. Vattenfall vastaa verkon suunnittelusta ja valvonnasta. Suomen malli urakoitsijayhtiöiden käytössä eroaa merkittävästi muusta Euroopasta, jossa on tavallista että rakentaminen ja kunnossapito hoidetaan verkkoyhtiöiden oman henkilöstön voimin. Tämän vuoksi myös järjestelmät asettavat tiettyjä haasteita, joista kerrotaan tarkemmin alakappaleessa 2.4.

Suomen nykyinen sähköverkko on rakennettu pääasiassa 70-luvulla ja se alkaa olla elinkaarensa päässä. Tämä aiheuttaa lähivuosina mittavia kustannuksia kaikille sähköverkkoyhtiöille, joiden tavoitteena on kaivaa uudet kaapelit maan alle. Nykyinen verk-

ko pohjautuu hyvin pitkälti maanpäällisiin ilmajohtoihin, jotka ovat alttiita erilaisten sääilmiöiden aiheuttamille vioille. Esimerkkinä vuoden 2010 elo-syyskuussa riehuneet neljä myrskyä aiheuttivat maanlaajuisesti kymmenien miljoonien vahingot muutaman viikon aikana. Kotitaloudet olivat pahiten kärsineillä alueilla ilman sähköä jopa viikon.

Myrskyt ovat ongelma etenkin Vattenfallille, koska sen verkkoalue eroaa merkittävästi ainoastaan kaupunkialueella toimivista verkkoyhtiöistä. Verkko levittäytyy todella laajalle ja kattaa pääasiassa haja-asutusalueita ja metsiä. Vaikeakulkuisen maaston vuoksi vikojen korjaaminen on toisinaan hidasta. Vika-alttiin verkon korjaus käynnistetään aina viiveettä, jotta asiakkaille voidaan turvata sähkön saanti kaikissa olosuhteissa.

2.4 Järjestelmät

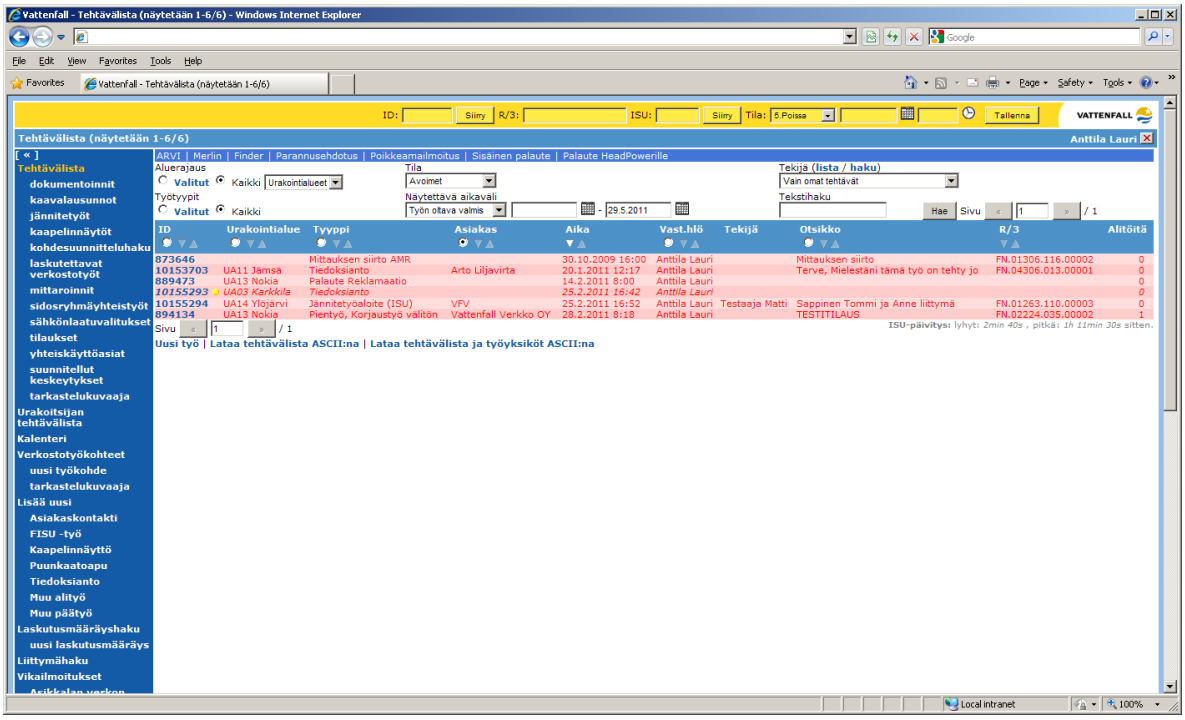
Vattenfallilla on käytössä useita muista toimialoista (esimerkiksi konepajateollisuudesta) poikkeavia järjestelmiä, joilla hallitaan sähköverkon eri komponentteja isoista sähköasemista kotitalouksien sähkömittareihin. Asiakaspalvelun järjestelmät eroavat teknisten toimintojen järjestelmistä, mutta kaikki ovat tavalla tai toisella integroitu keskenään.

Seuraavissa kappaleissa on kuvattu työnohjauksen kannalta olennaisimmat järjestelmät opinnäytetyön ja siihen liittyvän kehittämistehtävän alkuvaiheessa.

2.4.1 Avux -työnohjausjärjestelmä

Alun perin vuonna 2003 kehitetty ja käyttöönotettu Avux on asiakaskohtaisesti räätälöity web-pohjainen työnohjausjärjestelmä (kuva 1), jonka oli tarkoitus olla väliaikainen ratkaisu Vattenfallin nopean laajenemisen tuomiin haasteisiin. Se oli tuolloin ainoa järjestelmä, jolla hallittiin uusia rakentamisprojekteja, sähköliittymiä ja pientöitä. Toimittajana oli silloin nimellä Ajatuspaja toiminut pieni ohjelmistotalo, joka vuosien mittaan on kulkeutunut osaksi HeadPower Oy:tä.

Avux on järjestelmähkokonaisuus, joka on jaettu Intranet- ja Extranet-ympäristöihin. Intranet on integroitu SAP-järjestelmään ja se vaihtaa työtilaustietoja sen kanssa. Sisäisiä käyttäjiä järjestelmällä on n. 100. Intranet huolehtii myös XML (extensible markup language) -muotoisten tilausviestien lähettämisestä ns. sähköisen rajapinnan kautta urakoitsijoille ja niihin liittyvien kuittausten (”työ valmis” -viestien) vastaanottamisesta.



Kuva 1: Avuxin käyttöliittymä

Extranet-ympäristö on rakennettu urakoitsijoiden edustajille. Nämä voivat käydä tarkistamassa uudet työtilaukset ja kuitata ne suoraan Avuxiin. Urakoitsijalle on Avuxissa tarjolla myös melko merkittävät työnohjausmahdollisuudet, eli töitä voidaan kohdistaa työnjohtajan toimesta yksittäisille työntekijöille tai tiimeille. Käyttöliittymässä on jonkin verran eroja työnjohtaja- ja työntekijäprofiilien välillä. Extranet replikoi tietokantaansa reaaliajassa Intranetin kanssa. Aktiivisia käyttäjiä on n. 500.

Avux on siis täysin Vattenfallille räätälöity, alun perin terveydenhuollon tarpeisiin kehitetty ratkaisu, jota on vuosien varrella räätälöity yhä enemmän. Väliaikaisesta ratkaisusta kehittyi jossain vaiheessa työnhallinnan ”master-järjestelmä”, johon on liitetty monenlaisia toimintoja.

Avux on hyvä esimerkki siitä, miten Vattenfall Verkon sähköverkon rakentamis- ja kunnossapito eroaa yleisestä käytännöstä. Koska työnohjaus pohjautuu pääsiassa siihen, että tilaukset saadaan välitettyä urakoitsijoille, eikä niinkään oman henkilöstön hallintaan, ei markkinoilla ole ollut (eikä kirjoitushetkelläkään ole) ”hyllytavarana” tähän tarkoitukseen soveltuvaa työnohjausjärjestelmää. Kaikki markkinoilla olevat valmiit ratkaisut nojaavat enemmän tai vähemmän oman henkilöstön töiden aikatauluttamiseen ja asentajien seuraamiseen kentällä. Näin olleen on täytynyt turvautua epästandardiin ratkaisuun, joka pitkällä aikavälillä on osoittautunut haasteelliseksi hallita.

2.4.2 SAP

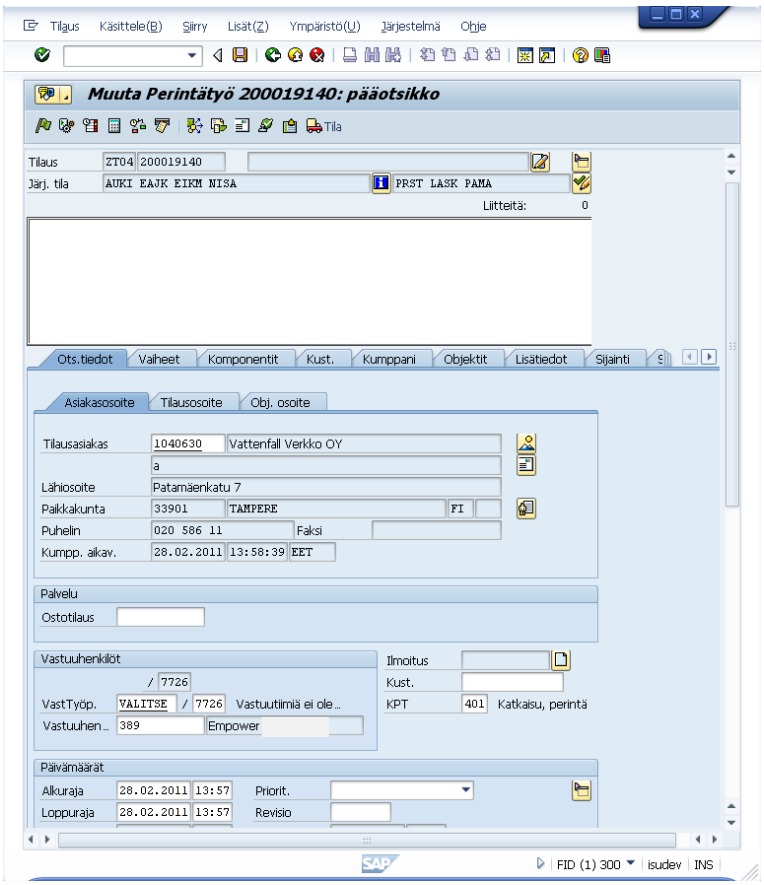
SAP on maailman johtava ERP-ratkaisujen toimittaja, jonka mainoslause ”The best run businesses run SAP” lupaa paljon. Vattenfallilla on käytössä lukuisia SAP-järjestelmiä jotka ovat sekä maa- että konsernikohtaisia. Suomessa on käytössä CRM (Customer Relationship Management)-, BW (Business Warehouse)- ja Utilities-versiot, joiden lisäksi Suomen toiminnot käyttävät Ruotsin kanssa yhteistä talousjärjestelmää. Vattenfall Verkon pääasialliset työkalut ovat CRM ja Utilities (myöhemmin ISU).

ISU on erityisesti sähköverkkoyhtiöiden toimintaa tukeva versio SAPista ja se pitää sisällään kaikki olennaiset tiedot asiakkaista, sähkömittareista ja kulutushistoriasta. Lisäksi sen avulla hallinnoidaan työtilauksia. Kuvassa 2 on esimerkki ISU:n käyttöliittymästä.

CRM on asiakkuudenhallintaversio SAPista, joka on integroitu tiiviisti ISUn kanssa. Se on asiakaspalvelun ensisijainen työkalu ja sen kautta tehdään muutoksia asiakkaiden tietoihin ja luodaan tarvittaessa palvelutilauksia. Tiedot synkronoidaan ISUn kanssa reaaliajassa.

Raportointi hoidetaan yleisesti ottaen BW -palvelimen kautta, joskin eri järjestelmät tuottavat perustason vakiomuotoisia raportteja itsenäisestikin.

Integraatiot eri SAP-asennusten ja muiden niihin liitettyjen järjestelmien välillä hoidetaan SAP XI-integraatiopalvelimen kautta. Data liikkuu pääasiassa SAPin omassa IDOC-muodossa, jota voidaan tehokkaasti tarkkailla sanomatasolla.



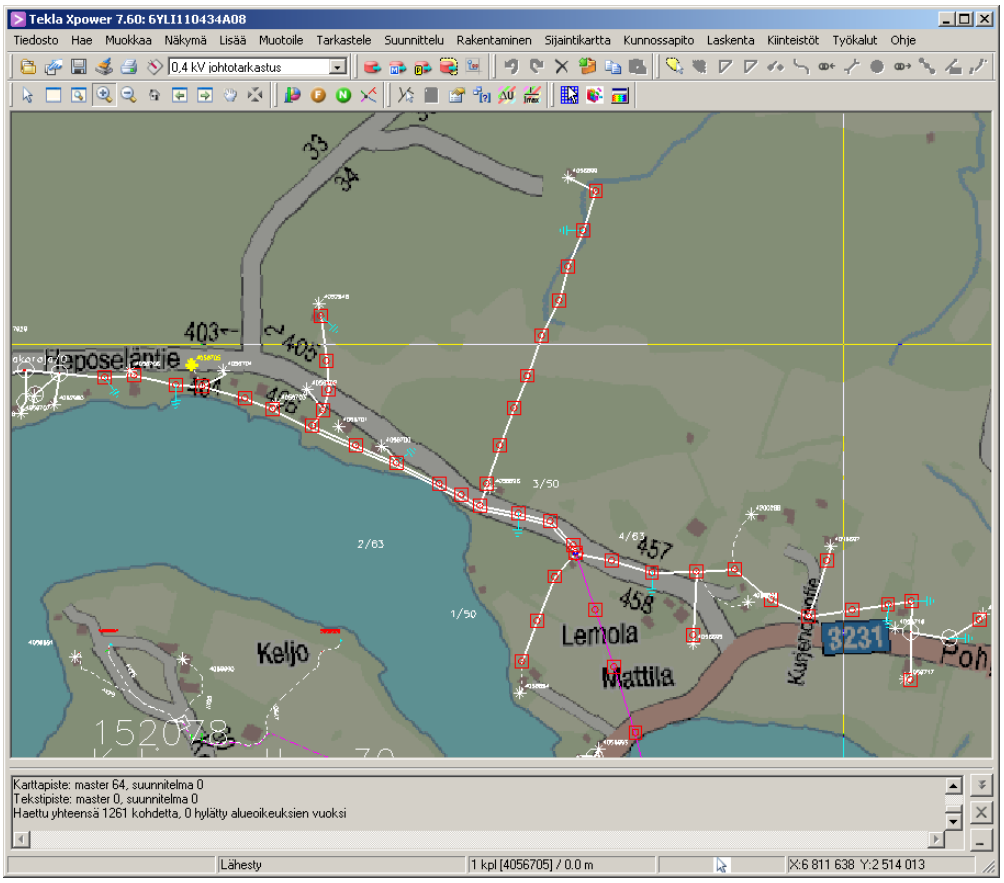
Kuva 2: SAP ISU:n käyttöliittymä (PM- eli, Plant Maintenance -moduuli), näkymä työtilauksen luomisesta

ISUa käytetään etenkin pientöiden, kuten perintä- ja mittarointitöiden luomiseen, kuin myös vikatilausten välittämiseen Tekla DMS-järjestelmästä eteenpäin. SAPin PM-moduulin piti alun perin korvata Avux työnohjauksen master-järjestelmänä, mutta rakentamisprojekteja ei missään vaiheessa siirretty hoidettavaksi sen puitteissa rakentamisprosessin tarpeista johtuvien syiden vuoksi.

2.4.3 Tekla NIS

Tekla NIS, entiseltä nimeltään Tekla Xpower, on ensisijaisesti verkko-omaisuuden dokumentointiin ja hallintaan tarkoitettu järjestelmä (kuva 3), joka pitää sisällään tiedot

Vattenfallin sähköverkon kaikista osa-alueista. Järjestelmää on laajennettu Tekla DMS -tuotteella, jolla pystytään kontrolloimaan sähkönjakelua.



Kuva 3: Tekla NIS:n käyttöliittymä

DMS:n kautta luodaan myös työtilauksia, jotka koskevat sähköverkon viankorjausta. Nämä työt ovat asiakkaan kannalta kriittisimpiä, koska kyseessä on miltei aina sähkön katkeamiseen liittyvä tehtävä. DMS lähettää uudesta vikatyöstä tiedon SAPiin, jossa siitä generoituu työtilaus PM-moduuliin ja välittyy sen kautta edelleen Avuxiin.

2.4.4 Välityspalvelu

XML-muotoisten tilausten toimittaminen Avuxista urakoitsijoiden järjestelmiin on hoidettu välityspalvelun kautta. Tämä tarkoittaa että Vattenfallilla ei ole urakoitsijakohtaisia point-to-point -yhteyksiä, vaan välityspalveluntarjoaja huolehtii sanomien ohjaamisesta oikeille vastaanottajille. Ylläpidollisesti ja tietoturvasyistä tämä on hyvä vaihtoeh-

to, koska uusia toimijoita voidaan ottaa mukaan tekemättä muutoksia tietoliikenneyhteyksiin tai palomuuriin. Palveluntarjoaja ohjaa viestit oikeaan osoitteeseen viestien sisältämän OVT-tunnuksen perusteella ja vastavuoroisesti toimittaa urakoitsijoiden vastausviestit takaisin Avuxiin.

3 TYÖN TAUSTA JA TAVOITTEET

Työn taustalla on kaksi toisistaan sinänsä riippumatonta asiaa, jotka tulivat ajankohtaiseksi yhtä aikaa. Ensimmäinen on Vattenfallin alkuun panema, suomalaisten sähköverkkoyhtiöiden yhdessä kehittämän XML-standardin käyttöönotto ja toinen Avux-järjestelmän elinkaaresta johtuva uudistamistarve. Yhdistämällä nämä kaksi asiaa, pystyttiin yhdellä kertaa päivittämään sekä sisäiset työtilausprosessit, että urakoitsijoiden käyttämä sähköinen työtilausten käsittely.

3.1 TIEKE-standardi

Ainakin isoimmat sähköverkkoyhtiöt ovat jo vuosia tilanneet osan tai kaikki rakentamis- ja pientyönsä urakoitsijoilta sähköisten rajapintojen välityksellä. Yhtiöiden omista järjestelmistä välitetään tilausviestit suoraan urakoitsijan työnohjausjärjestelmään, jolloin tilausprosessi nopeutuu ja automatisoituu. Ongelmana on kuitenkin ollut sanomamääritysten laaja kirjo. Käytännössä jokaisella verkkoyhtiöllä on oma, pahimmillaan jopa urakoitsijakohtainen standardinsa. Tämä asettaa suuria vaatimuksia tietojärjestelmille, joiden täytyy pystyä käsittelemään monen tyyppisiä sanomia ja muuntaa ne ymmärtämäänsä muotoon. Lisäksi uusien toimijoiden kytkeminen olemassa oleviin järjestelmiin on hidasta, koska integraatiot pitää rakentaa alusta asti uudestaan, dokumentaatio on vähäistä ja järjestelmien resurssit voivat olla puutteelliset.

Energiateollisuus ry:n vuonna 2009 käynnistämänä aloitettiin sähköverkkoyhtiöiden ja urakoitsijoiden yhteinen projekti, jonka tavoitteena oli määritellä yksi yhteinen XML-sanomastandardi työnohjausjärjestelmien väliseen sanomaliikenteeseen. Projektin vetovastuu annettiin Tietoyhteiskunnan kehittämiskeskus TIEKE ry:lle, jolla on kokemusta vastaavien standardien laatimisesta mm. kaupan alalle. Sähköverkkoyhtiöistä mukana Vattenfallin lisäksi olivat mm. Fortum Sähkönsiirto Oy ja Helen Sähköverkko Oy, kun taas urakoitsijapuolta edustivat mm. Empower Oy, Voimatel Oy ja Suomen Energia Urakointi Oy. Projekti päättyi kesäkuussa 2010 ja sen tuloksena syntyi 13 sa-

nomaa, joiden avulla pystytään hoitamaan osapuolten välinen työtilaus- sekä tarjousprosessi.

Uusi standardi mahdollistaa sen, että verkkoyhtiöt voivat integroida uusia urakoitsijoita järjestelmäänsä käytännössä välittömästi. Koska kaikki rajapinnassa toimivat yritykset käyttävät samoja viestejä, tarvitsee uusi toimija ainoastaan määrittää esimerkiksi OVT (organisaatioiden välinen tiedonsiirtotunnus)-numeronsa perusteella viestien vastaanottajaksi. Teknisiä muutoksia järjestelmiin ei siis enää vaadita siinä määrin kuin ennen. Urakoitsijayrityksille standardi puolestaan mahdollistaa sen, että ne voivat halutessaan laajentaa toimintaansa uusien verkkoyhtiöiden alueelle todella nopeasti, mikäli verkkoyhtiöllä on olemassa TIEKE-sanomien mukainen rajapinta.

Vattenfall oli projektissa aloitteentekijä ja yksi aktiivisimmista toimijoista ja teki jo projektin aikana periaatepäätöksen viestien käyttöönoton aloittamisesta heti, kun se vain on mahdollista. Tämä tarkoitti järjestelmäprojektin valmistelun aloittamista jo vuoden 2010 loppupuolella.

3.2 Avux-järjestelmän tekniset haasteet

Kuten jo aiemmin on todettu, Avux-järjestelmää ei koskaan tarkoitettu pitkäaikaiseen käyttöön. Vuonna 2003 se ajoi asiansa ja sopi silloisiin toimintamalleihin erinomaisesti. Koska järjestelmä pohjautuu jo kahdeksan vuotta vanhaan tekniikkaan ja on alun perin räätälöity toimimaan urakoitsijoiden suuntaan ainoastaan HTTP:n (hyper text transfer protocol) avulla, viime vuosina mukaan otetut XML-rajapinnat ovat tehneet kokonaisuudesta liian laajan. Web-palveluksi tarkoitettu järjestelmä ei täytä integraatiopalvelimelle asetettuja vaatimuksia ja viestinkäsittelylogiikka on alkeellista.

Vuosien mittaan räätälöidyt ominaisuudet ovat rönsyilleet ja muuttaneet tietokantarakenteen sekavaksi ja vaikeaksi tulkita. Olemassa olevat XML-rajapinnat noudattavat useaa eri viestimallia vastaanottajasta riippuen. TIEKE-standardin myötä edessä olisi

ollut joka tapauksessa Avuxin viestinkäsittelylogiikan muutos, mikä itsessään olisi ollut jo mittava investointi saatuun hyötyyn nähden.

Ylläpidollisesti järjestelmä on myös haastava, sillä se on ainoa laatuaan ja järjestelmän toimittajan tuki sovellukselle on loppumassa. Järkevämpää kuin Avuxin modernisointi, oli lähteä ajattelemaan toimintaa puhtaalta pöydältä ja ohjata toimintaa olemassa olevaan SAP-järjestelmään.

3.3 Worms-järjestelmä

Edellä kuvattujen seikkojen vuoksi Vattenfall Verkko alkoi tutkia vaihtoehtoisia ratkaisuja työtilausten hoitamiseksi ja niiden välittämiseksi urakoitsijoille. Koska yrityksellä oli jo osittain käytetty työnohjaukseen SAPia, oli se looginen suunta ohjata omaa toimintaa.

SAPin käytön ongelmaksi olisi kuitenkin muodostunut sen raskas ja kallis integrointi urakoitsijoiden järjestelmiin. Myös työnohjaukseen liittyvä raportointi olisi ollut kankeaa, koska se olisi nojannut SAPin BW:hen johon on äärimmäisen vaikea tuottaa uusia, etenkin ”ad hoc”-tyyppisiä raportteja ilman SAP-konsulttien apua.

Aiemmin Vattenfallissa oli ideoitu ns. ”magic boxia”, joka ratkaisisi nämä ongelmat. Markkinoilta ei vain löytynyt valmista tuotetta, joka olisi soveltunut juuri tähän tarkoitukseen. Tämän vuoksi oli hankittava räätälöity ohjelmisto ja määritellä sille juuri sellaiset ominaisuudet, kuin Vattenfall tarvitsi. Syntyi ”Worms”, joka on lyhenne sanoista Work Order Management System. Järjestelmä kytkettäisiin SAPiin käyttämällä SOAP (simple object access protocol)-pohjaista integraatiota ja se toimisi työtilausten tietovarastona ja välityskanavana urakoitsijoiden suuntaan.

SAPissa luodut työt välittyvät reaaliajassa Wormsin tietokantaan, josta voidaan koostaa monipuolisia raportteja. Urakoitsijoille työt välittyvät yhtälailla reaaliajassa joko TIEKE-sanomina tai kevyen web-käyttöliittymän kautta. Urakoitsijat puolestaan rapor-

toivat töiden edistymisestä Wormsin suuntaan, joka siirtää muutokset SAPiin. Käyttöönotto on suunniteltu tapahtuvaksi vuoden 2012 alussa.

3.4 Tavoitteet

Opinnäytetyön tavoitteena on tehdä kattava järjestelmätestaus Vattenfall Verkko Oy:n uudelle Worms-järjestelmälle ennen sen käyttöönottoa. Testaus käsittää FURPS-mallin (kts. luku 4.3) määrittämät osa-alueet ja se tehdään osittain koneellisesti.

Testauksen aikana etsitään virheitä ja suorituskykyongelmia Wormsista simuloimalla mahdollisimman tehokkaasti erilaisia käyttötapauksia ja kuormaa. Havaitut kriittiset poikkeamat laadussa korjataan ennen käyttöönottoa. Mahdolliset testauksen aikana esiin nousseet lisäominaisuudet ja muut, toiminnan kannalta epäolennaiset puutteet arvioidaan tapauskohtaisesti ja korjataan tarpeen mukaan heti tai myöhemmässä päivityksessä.

Lopputuloksena on vaatimusmäärittelyn mukainen ja toimintavarma järjestelmä, joka kestää suurta kuormitusta ja on sopeutuvainen muuttuviin toimintaolosuhteisiin. Vattenfallille projektista syntyy kattava testi- ja toiminnallisuusdokumentaatio, jonka avulla järjestelmän toimintaa voidaan tarkastella tulevaisuudessa ja hyödyntää jatkokehityksessä.

3.5 Rajaukset

Tämä opinnäytetyö keskittyy Vattenfall Verkko Oy:n Worms-järejstelmän suorituskyky- ja toiminnallisuustestaukseen järjestelmän rakennusvaiheessa.

Kokonaisuutena kehittämistehtävän tuloksena syntyy Vattenfall Verkko Oy:lle uusi tapa luoda ja välittää työtilauksia, käsittäen urakoitsijarajapinnan lisäksi SAP-järjestelmän erilaiset muutokset, SAPin ja Avuxin välisen SOAP-rajapinnan sulkemisen,

oman henkilökunnan koulutuksen ja uusien raporttien rakentamisen Wormsin päälle rakentuvaan tietovarastoon. Nämä asiat on kuitenkin rajattu opinnäytetyön ulkopuolelle.

4 OHJELMISTOKEHITYKSEN JA -TESTAUKSEN PERIAATTEET

Ohjelmistotestausta ei voida pitää omana, muusta kehitystyöstä riippumattomana kokonaisuutena. Se ei toisaalta ole myöskään puhtaasti kehitystyötä, vaan kategorisoituu tukitoiminnoksi. Testaaminen on tärkeä osa ohjelmiston elinkaarta aina kehitysideasta käyttöönottoon ja lopulta käytöstä poistoon. (Hass 2008, 1)

Seuraavissa alaluvuissa käydään läpi ohjelmistokehityksen eri malleja ja keskitytään sen jälkeen itse testaamisen ja laadunvarmistuksen tärkeyteen ohjelmistoprojektissa.

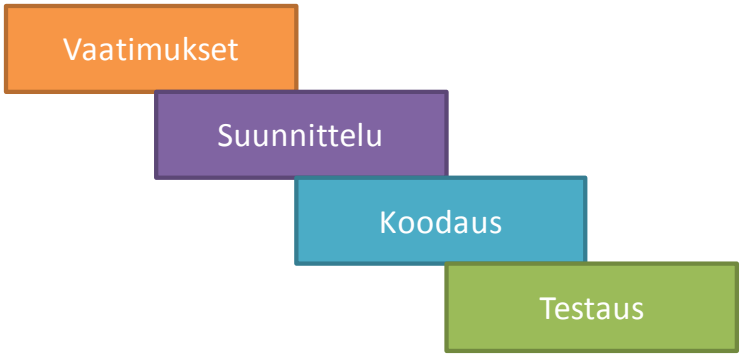
4.1 Ohjelmistokehityksen eri mallit

Ohjelmistojen rakentaminen koostuu neljästä peruspilarista, jotka ovat vaatimusmäärittely, suunnittelu, ohjelmointi ja testaus. Näistä muodostuu useaan pääkategoriaan jaettavia ohjelmistokehitysmalleja, jotka ovat sekventiaalinen, sekä iteratiivinen ja inkrementaalinen.

4.1.1 Sekventiaalinen malli (Sequential model)

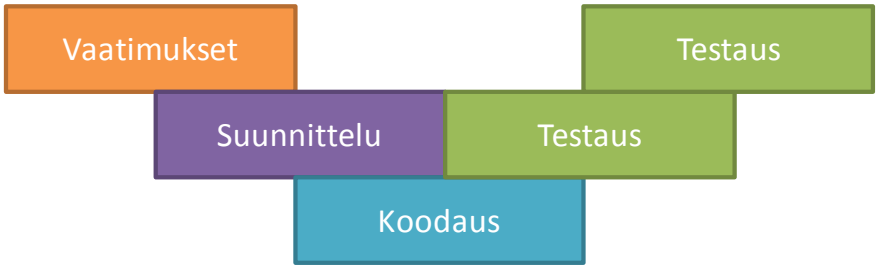
Sekventiaalisen mallin lähtökohtana pidetään sitä, että asiakas tietää mitä hän ohjelmistoltaan haluaa, on jäädyttänyt vaatimukset (muutokset poikkeuksia) ja järjestää katselmuksia tietyissä projektin vaiheissa, joissa annetaan palautetta

Mallin onnistuneen läpiviennin kulmakivet ovat stabiilit vaatimukset ja ympäristö, painopiste kokonaiskuvassa ja yksi lopullinen toimitus. Sekventiaalisen mallin pohjana on ns. vesiputousmalli, jossa vaiheet jäsentyvät nimensä mukaisesti ”vesiputoukseksi” (kuva 4) päättyen testaukseen projektin loppuvaiheessa. (Hass 2008, 3)



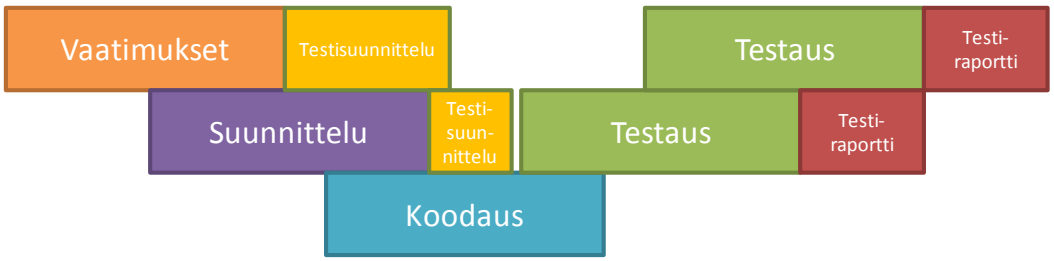
Kuva 4. Vesiputousmalli

Vesiputousmallista jalostetumpi versio on ns. V-malli, jossa testausta on enemmän. Nopeasti vilkaistuna (kuva 5) voi saada käsityksen, että tässäkin mallissa testaaminen jätetään projektin loppupuolelle. Näin ei kuitenkaan ole.



Kuva 5. V-malli

Todellisuudessa V-mallia hyödynnetään hieman tehokkaammin. Prosessin selkeyttämiseksi on olemassa erillinen W-malli, joka kuvaa tarkemmin miten testaus hallitaan (kuva 6). Käytännössä malleilla ei ole esitystavan lisäksi mitään eroa. Mallissa vaatimusmäärittely tehdään kuten vesiputousmallissakin, mutta heti sen perään aletaan suunnitteluvaiheen rinnalla rakentaa testisuunnitelmaa, jonka pohjalta tuote testataan. Testisuunnitelmaa jalostetaan suunnitteluvaiheen jälkeen, kun varsinainen ohjelmointi on käynnissä. Näin ollen samanaikaisesti voidaan määritellä paitsi se mitä tuotteen pitäisi tehdä, mutta myös kuinka toiminnallisuus voidaan testata tehokkaasti. (Hass 2008, 4-5)



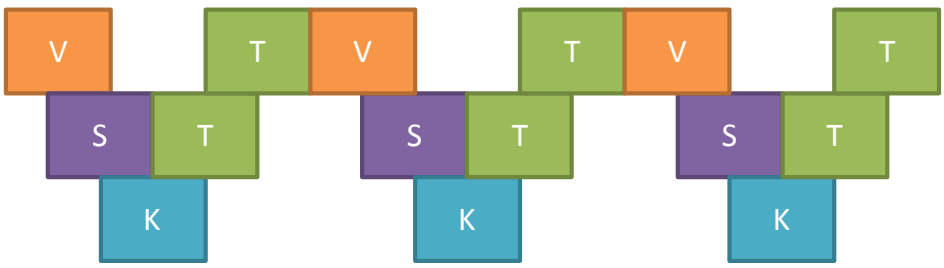
Kuva 6. W-malli

4.1.2 Iteratiivinen ja inkrementaalinen malli (Iterative & incremental models)

Iteratiivisen ja inrementaalisen mallin mukaista ohjelmistokehitystä käytetään esimerkiksi Scrum-projekteissa. Scrumista kerrotaan lisää seuraavassa kappaleessa. Käytännössä tämä malli koostuu useista sekventiaalisen mallin mukaisista jaksoista, joiden lopputuloksena syntyy ohjelmisto.

Lähtökohta tälle mallille on se, että asiakas ei pysty täysin kuvaamaan tarvitsemaansa sovellusta, vaatimuksiin on odotettavissa muutoksia ja tuotetta arvioidaan jatkuvasti ja siitä annetaan palautetta

Iteraatiot ovat vaiheita, joiden aikana ei anneta tuotosta asiakkaan testattavaksi ja inkrementaatiot koostuvat useammasta ohjelmiston alikomponentista, jotka voidaan luovuttaa asiakkaan testattavaksi iteraation jälkeen.



Kuva 7. Iteratiivinen ja Inkrementaalinen kehitysmalli

Onnistuneen I&I –mallin mukainen projekti vaatii

- nopeaa ja jatkuvaa palautetta asiakkaalta
- mukautuvaa lopputuotteen määritelmää
- keskittymistä tärkeimpiin ominaisuuksiin
- jatkuvasti päivittyviä testiversioita järjestelmästä

Mallin käyttö on tyypillistä projekteille, joissa ”asiakas ei tiedä mitä haluaa, kunnes näkee sen”. Tilanteesta on olemassa epävirallinen lyhenne IKIWISI: ”I Know It When I See It”. (Hass 2008, 5-6)

Tätä mallia käytetään myös tässä opinnäytetyössä, koska ohjelmistotoimittaja käyttää projektinhallinnan välineenä Scrumia.

4.1.3 Scrum ohjelmistoprojektin- ja laadunhallinnan työkaluna

Scrum on ketterä (eng. agile) projektinhallintamalli, jossa huomiota kiinnitetään enemmän tekemiseen ja muutoksiin varautumiseen kuin alkuvaiheen tarkkaan suunnitteluun. Mallin kolme pääperiaatetta ovat läpinäkyvyys, työn tulosten säännöllinen tarkasteleminen ja työn sopeuttaminen toleransseihin (Lekman 2011). Nämä pääperiaatteet löytyvät myös vuonna 2001 julkaistusta Agile Manifestosta, johon Scrum vahvasti nojaa.

Manifestin olennaisimmat periaatteet ovat:

- Ihmiset ja vuorovaikutus ennen prosesseja ja työkaluja
- Toimiva ohjelmisto ennen kattavaa dokumentaatiota
- Asiakasyhteistyö sopimusneuvottelujen sijaan
- Muutoksiin sopeutuminen tiukan suunnitelman seuraamisen sijaan

(Agilemanifesto.org 2001)

Se että jotakin asiaa arvostetaan enemmän kuin toista, ei kuitenkaan tarkoita että esimerkiksi dokumentaatio pitäisi unohtaa kokonaan. Lauseilla lähinnä painotetaan sitä, että ohjelmistoprojektin lähtökohta on tuottaa asiakkaan tarpeet täyttävä, toimiva ohjelmisto, jota toissijaisesti tuetaan dokumentaatiolla ja muilla välineillä.

Scrum-mallissa projektiryhmä koostuu toimittajan scrummasterista (Scrum master) ja itseohjautuvista tiimeistä (Scrum team) sekä tilaajaosapuolen tuoteomistajasta (product owner). Ohjelmiston vaatimukset kootaan tuotteen kehitysjonoon (product backlog), joka toimii korkean tason työlistana.

Tuoteomistajan tehtävä on ylläpitää ja priorisoida kehitysjonon tehtäviä projektin aikana. Uusia ominaisuuksia voidaan ottaa mukaan ja vanhoja pudottaa pois, mikäli ne eivät täysin muuta projektin laajuutta, eli scopea (Pham 2011, 6-10). Tältä osin Scrum eroaa radikaalisti vesiputousmallista, jossa muutoksia pyritään välttämään viimeiseen asti. Scrum lähtee oletuksesta, että muutoksia tulee varmasti projektin aikana.

Varsinainen työ pilkotaan yleensä n. 1-2 viikon mittaisiin sprinttehiin (sprint), joille määritellään tietty määrä tehtäviä, eli tarinoita (story) tuotteen kehitysjonoa suppeampaan sprintin tehtävälistaan (sprint backlog). Sprintin tavoitteena on tuottaa asiakkaalle ”toimiva versio” tuotteesta, eli käytännössä jotain esiteltävää ja testattavaa (ei siis välttämättä kokonaista järjestelmää). Sprintti aloitetaan suunnittelukokouksella (sprint planning) ja päätetään katselmointiin (sprint review). Sprintin aikana pidetään päivittäisiä Scrumeja (daily Scrum), joissa tiimi jakaa töitä kyseiselle päivälle ja tarvittaessa ilmoittaa havaituista ongelmista scrummasterille, jonka tehtävä on ratkaista ongelmat tavalla tai toisella. (Schwaber, Sutherland 2010, 9-14)

Sprintiin valittavat tarinat pisteytetään tarinapisteillä (story point), jotka eivät suoranaisesti kuvaa tehtävän viemää aikaa tunneissa tai päivissä, vaan enemmänkin kuvastavat ohjelmoijan arviota työn laajuudesta. Pisteiden vaihteluväli on esim. 1-40, joista 1 on erittäin pieni tehtävä ja 40 niin suuri kokonaisuus, ettei sitä pysty käsittelemään (esim. puutteellisten tietojen tai mittavan toiminnallisuuden vuoksi). 40 pisteen tehtävät jaetaankin yleensä pienemmiksi kokonaisuuksiksi, jolloin ne on helpompi pisteyttää.

Pistetys tehdään ohjelmoijien kesken pelaamalla ”suunnittelupokeria” – Planning Pokeria – jonka tarkoituksena on löytää tiimin mielestä yhteneväinen arvio yhden tehtävän työmäärästä. Ohjelmoijat nostavat omasta korttipakastaan mielestään sopivan

arvion työmäärästä ja ne käännetään yhtä aikaa kaikkien nähtäville. Mikäli työmäärät eroavat eri ohjelmoijien välillä huomattavasti, tulee heidän perustella arvionsa muille (Wikipedia 2011).

Sprintin aikana seurataan kehityksen nopeutta (velocity). Sillä pystytään arvioimaan ehditäänkö tehtävät suorittaa sovitussa ajassa ja samalla arvioida valmiiksi paljonko työtä voidaan seuraavalla sprintillä tehdä (Pham 2011, 60-62). Tulevia sprintin tehtävälistalle valittavia tehtäviä voidaan nopeuden perusteella arvioida tarkemmin, sillä projektin alkuvaiheen jälkeen saadaan tuntuma siitä, paljonko tarinapisteitä tiimi pystyy Sprintin aikana todellisuudessa tekemään.

4.1.4 Scrum ja Worms-järjestelmä

Worms rakennettiin ohjelmistotoimittaja Futurice Oy:n toimesta käyttäen projektitoiminnan pohjana Scrumia. Ohjelmointi tehtiin pääasiassa 1-2 viikon mittaisissa sprinteissä, joille määriteltiin aina tietty teema ja tavoitteet. Esimerkiksi yksi teema oli ”Tilakone” ja sprintin kesto kaksi viikkoa. Sprintin tuloksena oli Apache Camelin päälle rakennettu käsittelysäännöstö, jolla hallittiin SAPista saapuvien tilaussanomien ohjaus oikeisiin työkulkuihin erilaisten muuttujien avulla. Keskimääräinen nopeus kahden ohjelmoijan tiimille oli 20 pistettä viikossa. Ohjelmoinnin loppuvaiheessa pidettiin myös todella lyhyitä (2-3 päivän mittaisia) sprinttejä, joiden aikana korjattiin bugeja.

Koska Scrum vaatii tilaajan (tässä tapauksessa Vattenfall) edustajalta (tuoteomistaja) tiivistä yhteistyötä ohjelmistotoimittajan kanssa (ja se pystyttiin projektin aikana myös takaamaan), ohjelmisto voitiin sprinttien aikana rakentaa tehokkaasti vastaamaan Vattenfallin vaatimuksia. Mahdolliset ongelmat ratkaistiin ennen Sprintin loppumista jatkuvalla kommunikaatiolla Vattenfallin ja Futuricen välillä. Näin järjestelmästä saatiin aina jotakin valmiiksi sen sijaan, että Sprintin jälkeen olisi tutkittu puutteita ja korjattu niitä seuraavan Sprintin aikana. Havaitut pienemmät bugit, jotka eivät vaikuttaneet kokonaisuuden testaamiseen, korjattiin erillisessä bugi-sprintissä kehitystyön loppupuolella.

Scrumin käyttö loi haasteita Vattenfallin sisäiselle projektiorganisaatiolle, koska konsernin mallit pohjautuivat pitkälti vesiputousmalliin ja projektiryhmä oli tottunut siihen. Testaukseen liittyen oletus oli, että jo projektin alussa testivaihe olisi ollut aikataulutettu Gantt-kaavioon. Scrumin testaus pohjautuu puolestaan siihen, että testataan aina, kun testattavaa on.

4.2 Miksi panostaa ohjelmiston laatuun?

Forrester Researchin mukaan niillä yhdysvaltalaisilla yrityksillä, joiden ohjelmistoilla on alhaiset laatuvaatimukset, kului vuonna 2009 keskimäärin 39 % IT-budjetista ja 46 % työajasta väärin toimivien ohjelmistojen kanssa taisteluun. Kyseiset yritykset saavat hoidettua vähemmän projekteja valmiiksi ajallaan ja budjetissa, kuin paremmin laatuun panostavat kilpailijansa. (Gerush 2010, 3)

Huono ohjelmistojen laatu heijastuu paitsi asiakkaisiin, myös sisäiseen toimintaan. Väärin toimivat ohjelmistot turhauttavat käyttäjiä ja ohjaavat nämä hoitamaan asiat ”epästandardeilla tavoilla”, jolloin prosesseihin perustuvan organisaation tasaiseen laatuun panostaminen kärsii.

Tietojärjestelmien laatuun panostaminen nähdään eri tavalla yrityksen eri portaissa. Toimitusjohtaja voi sanoa, että yritys investoi huomattavia summia tietojärjestelmien kehittämiseen, kun IT-päällikön näkökulmasta valtaosa budjetista kuluu vanhojen ja toimimattomien ohjelmistojen ylläpitoon (Hannula 2010, 8). Kun ohjelmisto ei toimi, aiheuttaa se myös valtavia haamukuluja organisaatiossa, koska työt seisovat pahimmillaan sadoilla käyttäjillä. Toimimaton järjestelmä tarkoittaa mahdollisesti myös menetettyä bisnestä sekä maineen kärsimistä.

Paitsi näkemyserot laatuun panostamisessa, myös itse laatu nähdään monella eri tavalla sidosryhmästä riippuen. Käyttäjän mielestä laatu on toimintavarmuutta ja helppo-

käyttöisyyttä, IT-tuen mielestä ylläpidon helppoutta, bisneskumppanien näkökulmasta saavutettavuutta ja helppoa integroitavuutta. (Gerush 2010, 11)

Vaikka Forresterin tutkimuksesta voisi päätellä, että ohjelmistojen laatu unohtuu yrityksiltä luvattoman usein, ei tilanne kuitenkaan näytä olevan aivan niin hälyttävä. Muutos parempaan on havaittavissa Microsoftin Mext Oy:llä teettämässä tuoreessa tutkimuksessa, jonka perusteella loppukäyttäjäorganisaatiot Suomessa ovat suhteellisen tyytyväisiä ohjelmistoprojekteihinsa. Vastaajista valtaosa on panostanut merkittävästi omia resurssejaan projektien sujuvuuden varmistamiseksi. (Mext 2010)

Vastaajista yli puolet piti ohjelmistoprojektien laatua hyvänä tai odotusten mukaisena. Tulos on kuitenkin sinänsä ristiriitainen, sillä ainoastaan 37 % vastaajista totesi järjestelmän virheettömyyden olleen odotusten mukaisella tai sitä paremmalla tasolla. Virheitä siis esiintyy, mutta ne tunnutaan hyväksyvän muiden vaatimusten täyttäessä odotukset. (Mext 2010)

Testaaminen ja laadunvarmistus itsessään ovatkin yhä lapsenkengissä. Testaaminen ulkopuolisilla tahoilla on olematonta ja pääasiassa se hoidetaankin itse. Resursseja ei kuitenkaan kohdenneta riittävästi testaamiseen, vaan ne kulutetaan enemmänkin esimerkiksi projektinhallintaan. Ainoastaan 15 % vastanneista yrityksistä nimesi projektille aina oman vastuuhenkilön laadunvarmistukseen. (Mext 2010)

Tutkimuksen tekijän yhteenvedossa todetaan, että ohjelmistojen laatua saataisiin asiakasyritysten toimesta parannettua panostamalla vaatimusmäärittelyn ja testaamisen menetelmiin, jolloin ostaja saisi haluamansa tarkemmin, taloudellisemmin ja vähemmin viivästyksin. (Mext 2010)

4.3 FURPS-malli

Ohjelmiston laatu heijastuu suoraan koko yrityksen laatuun. Virheiden lukumäärä ja esiintymistiheys on suoraan verrannollinen siihen miten paljon henkilöstö voi saada aikaan.

Tietojärjestelmäyritys Hewlett-Packard kehitti jo 1970-luvulla ns. FURPS-mallin, jonka avulla voidaan analysoida järjestelmän laatua eri näkökulmista:

- toiminnallisuus (Functionality)
- käytettävyys (Usability)
- käyttövarmuus (Reliability)
- suorituskyky (Performance)
- huollettavuus (Servicability, Supportability)

(Eeles 2005)

Mallia jalostettiin vuonna 1992 eteenpäin HP:n työntekijän Robert Gradyn toimesta, jolloin syntyi käsite FURPS+. Plus-merkin tarkoituksena on kiinnittää ihmisten huomio alkuperäisten asioiden lisäksi myös seuraaviin asioihin:

- Järjestelmän rakennevaatimukset (Design requirements)
- Käyttöönoton kannalta olennaiset vaatimukset (Implementation requirements)
- Muiden järjestelmäkomponenttien huomioiminen (Interface requirements)
- Fyysiset rajoitteet (Physical requirements)

(Eeles 2005)

Opinnäytetyössä FURPSia sovelletaan sen kaikilta osin, keskittyen pääasiassa sen alkuperäisiin teemoihin, mutta myös hieman sivuten plus-mallin lisähuomiota vaativat osa-alueet. Malli antaa jo otsikkotasolla hyvän kuvan siitä, mitä kaikkea testauksessa pitäisi huomioida, mutta on syytä paneutua eri alueisiin vielä tarkemmin ja selvittää mitä ne sisältävät. Nämä alueet käydään läpi seuraavissa alakappaleissa.

4.4 Toiminnallisuustestaus

Toiminnallisuustestauksesta voidaan puhua myös komponenttitestauksena. Tietojärjestelmä koostuu tietystä määrästä ohjelmistokomponentteja, joille jokaiselle on määritelty tietty toiminnallisuus. Komponentit testataan ensin erikseen ja lopulta yhdessä. Tavoitteena on löytää poikkeavuudet komponenttien toiminnasta (Hass 2008, 10-11).

Terminä voidaan käyttää myös Black Box -testausta. Järjestelmään annetaan määritetty syöte (input) ja sille on määritelty toivottu tulos (output). Varsinainen toiminnallisuus tapahtuu ”mustassa laatikossa”, eli järjestelmän koodissa. Testaajan ei tarvitse tietää koodista mitään, vaan ainoastaan olla tietoinen siitä miten järjestelmän tulisi toimia. Black Box -testaus soveltuu paremmin toiminnallisten kuin ei-toiminnallisten seikkojen testaamiseen. Poikkeamat toiminnallisuuksissa raportoidaan järjestelmätoimittajalle.

Toiminnallisuudet kategorisoidaan siis sekä toiminnallisiin, että ei-toiminnallisiin:

Toiminnalliset	Ei-toiminnalliset
• Tiedon syöttäminen	• Tietoturva
• Tiedon muokkaaminen	• Käytettävyys
• Hakutoiminnot	• Vasteajat, suorituskyky
• Integraatiot	• Varmuuskopiot
• Käyttäjähallinta	• Dokumentaatio
• Jne.	• Jne.

(Gerush 2010, 8)

Vaikka integraatiot sisällytetään yllä kuvatussa taulussa toiminnallisiin vaatimuksiin, käsitellään niitä kuitenkin FURPS+ -mallissa erillisenä osana (interface requirements). Teknisesti voidaan kysyä mitä eroa toiminnallisella vaatimuksella ja liittymisvaatimuksella on? Itse käsitän sen ainakin siten, että toiminnallisessa testauksessa kiinnitetään huomiota integraation toimivuuteen testattavan järjestelmän näkökulmasta, kun taas liittymävaatimuksilla keskitytään siihen miten integraatioiden toisessa päässä olevat järjestelmät käsittelevät tietoa.

Toiminnallisen ja ei-toiminnallisen rajaa saadaan hämärrettyä esimerkiksi tietokantaan liittyvän tietoturvallisuuden osalta. Web-käyttöliittymässä voi olla hakukenttä, joka tekee kyselyitä tietokannasta SQL-hakuina. Mikäli tietoturvaa ei ole hakuscriptiä rakennettaessa huomioitu, on käyttäjän mahdollista tahtomatta tai tahallisesti väärinkäyttää hakukenttää ns. SQL Injectionin muodossa. Käyttäjä voi syöttää tietyn tyyppisen haun, joka suorittaakin aivan muun tietokantatoiminnon kuin oli tarkoitettu. Käyttäjä pystyy teoriassa ja myös käytännössä tuhoamaan tietokantatauluja tai hakemaan tietoa tauluista, joita ei ole tarkoitettu kaikkien saataville. Näin ollen testatessa hakukentän toiminnallisuutta, testataan samalla myös tietoturvaa. Tätä asiaa käydään paremmin läpi alakappaleessa 6.2.2, jossa Worms-järjestelmälle suoritetaan virheensietotestejä.

4.5 Käytettävyystestaus

Käytettävyys koostuu FURPS-mallissa ensisijaisesti siitä, että käyttäjät saadaan sitoutettua järjestelmän käyttöön. Tämä tarkoittaa siis sellaisen käyttäjäkokemuksen tarjoamista, ettei käyttäjä haikaile vanhojen järjestelmien perään. Käytettävyyttä arvioitaessa voidaan käyttää esimerkiksi perinteistä ajanottoa: Kuinka kauan tietyn asian suorittaminen kestää? (Leffingwell 2010, 343-344)

Käytettävyyteen voidaan yhdistää neljä peruselementtiä:

- nopeus (performance)
- tarkkuus (accuracy)
- muisti (recall)
- käyttäjäpalaute (emotional response)

Käyttäjänäkökulmasta katsottuna järjestelmä täyttää onnistumisen kriteerit jos käyttäjän tarvitsema tieto on nopeasti ja helposti saatavilla ja toimenpiteiden suorittaminen järjestelmän sisällä on loogista ja yksinkertaista.

Wormsin web-käyttöliittymän sekä sähköisen rajapinnan tapauksessa voidaan melko helposti vertailla erilaisia vasteaikoja Avuxiin: Kauanko tilausviestin välittäminen Vattenfallin järjestelmästä urakoitsijan järjestelmään kestää? Kuinka kauan web-

käyttöliittymän aloitussivu latautuu kirjautumisen jälkeen? Avuxin kohdalla urakoitsija on usein törmännyt hitauteen sekä siihen, että järjestelmä ei toimi oletetulla tavalla. Kun nämä ongelmat saadaan ratkaistua uuden järjestelmän avulla, vähentää se myös Vattenfallin suuntaan tapahtuvaa reklamointia järjestelmän epävakaudesta.

4.6 Käyttövarmuustestaus

Käyttövarmuudella tarkoitetaan lähtökohtaisesti sitä, kuinka hyvin järjestelmä pysyy toimintakykyisenä. Useimmiten tavoitearvona on 99,9 % saavutettavuus esimerkiksi tietyn ajanjakson aikana (työpäivä, klo 8-23 tms.). Tämä tarkoittaa että järjestelmään ei saisi pääsääntöisesti tulla minkäänlaisia ongelmia silloin, kun sitä aktiivisesti käytetään. (Leffingwell 2010, 344)

Käyttövarmuutta voidaan testata etenkin erilaisilla kuormatesteillä, joilla pyritään selvittämään palvelimen kyky suoriutua oletetusta päivittäisestä kuormasta sekä mahdollisista piikeistä. Nämä testit liittyvät hyvin läheisesti suorituskykytestaukseen (alaluku 4.7). Mikäli suorituskyky pettää, häviää samalla käyttövarmuus.

Käyttövarmuuteen liittyy myös turvallisuusnäkökulmat, esimerkiksi palvelunestohyökkäyksiin reagointi sekä virustorjunta. Wormsin tapauksessa palvelin sijaitsee yhtiön sisäverkossa paikassa, johon ei ole mahdollista suoraan kohdistaa hyökkäyksiä, joten testauksen osalta varmuuskopiointiin, virustorjuntaan ja käyttäjähallintaan liittyvät asiat eivät sisälly opinnäytetyön käsittämän testauksen piiriin.

4.7 Suorituskykytestaus

Suorituskykyä mitatessa huomio pitäisi keskittää vaste- ja läpimenoaikoihin, käyttäjämäärien hallintaan, skaalautuvuuteen ja resurssien käyttöön (Leffingwell 2010, 345).

Jotta suorituskykyä voidaan mitata, tulee eri toiminnoille voida asettaa tavoitearvoja. Paljonko järjestelmällä on yhtaikaisia käyttäjiä? Miten paljon sanomia järjestelmien välillä kulkee? Siirretäänkö järjestelmässä isoja tiedostoja? Kun ohjeelliset arvot on määritelty, voidaan testata suoriutuuko järjestelmä työstään annettujen rajojen puitteissa.

Testauksessa pitäisi myös kiinnittää huomiota siihen, miten järjestelmä suoriutuu mahdollisesti tulevaisuudessa kasvavista vaatimuksista. Jos käyttäjämäärä kaksinkertaistuu, suoriutuuko järjestelmä kuormasta ilman päivityksiä?

Mikäli ohjelmistoprojektin aikana uudistetaan (kuten Wormsin tapauksessa) jo olemassa olevaa järjestelmää, pitäisi yrityksellä olla hyvä kuva esimerkiksi käyttäjämääristä. Näitä tietoja voidaan käyttää uuden järjestelmän testauksessa hyvänä pohjatietona, jolloin testauksessa käytettävä kuorma vastaa paremmin todellisuutta. Mikäli järjestelmä on kokonaan uusi, tulisi käyttäjämäärät pystyä arvioimaan mahdollisimman hyvin.

Wormsin osalta avaintunnuslukuja ovat tilausten siirtyminen SAPin suunnasta tulevasta SOAP-muodosta TIEKE-määritysten mukaisiksi XML-sanomiksi (ja toisinpäin) sekä jo aiemmin käytettävyyden yhteydessä mainittu web-käyttöliittymän nopeus erilaisten toimintojen osalta. Teknisesti järjestelmä on rakennuttu hyvin kevyeksi ja suoraviivaiseksi, mutta se tulee testivaiheessa asettaa äärirajoille välittämällä sille suuria määriä tietoa eri kanavia pitkin.

4.8 Huollettavus

Huollettavuudella tarkoitetaan sitä, miten helposti järjestelmää voidaan päivittää ja huoltaa. Huomiota tulisi jo järjestelmän rakennusvaiheessa keskittää sellaisiin ominaisuuksiin, jotka saattavat todennäköisesti muuttua tulevaisuudessa (esim. veroprosentit, tietoliikenneprotokollat)(Leffingwell 2010, 345). Wormsin osalta tällaisia arvoja ovat esimerkiksi integraatioihin liittyvät palvelinten osoitteet sekä TIEKE-sanomien

sisällä välitettävät Vattenfallin yhteystiedot. Jos siis esimerkiksi Vattenfall Verkon päätoimipiste muuttaa, pitää sen osoitetietoja pystyä helposti muokkaamaan järjestelmässä.

Ylläpidon kannalta on olennaista, että järjestelmälle löytyy tukea sekä laitteisto- että ohjelmistopuolelta. Liian monimutkaisesti toimiva järjestelmä on vaikea päivittää ja kallis ylläpitää. Huoltokatkot vievät paljon aikaa, mikä tarkoittaa palveluiden saavuttamattomuutta käyttäjille.

Laitteistorikon sattuessa on olennaista se, onko varaosia saatavilla ja miten esimerkiksi varmuuskopiot saadaan palautettua ja järjestelmän toimintakyky yleisesti palautettua, jos palvelimen kiintolevyt hajoavat.

4.9 Liittymä-/integraatiotestaus

Integraatiotestauksella selvitetään toimivatko järjestelmien väliset liittymät siten, kuin on tarkoitettu. Liikkuuko data oikeassa muodossa ja käsittelevätkö järjestelmät sisältöä oikein?

Mahdollisia tapoja testata integraatioita ovat esimerkiksi:

- top-down (ylhäältä alas)
- bottom-up (alhaalta ylös)
- functional integration (toiminnallinen testaus)
- big bang (kokonaisuustestaus)

Ylhäältä alas -mallissa testataan integraatiot vaihe vaiheelta aloittaen ”ydinjärjestelmästä” siirtyen sitten kohti ulompia osia. Vastavuoroisesti alhaalta ylös -mallissa aloitetaan ”reunoilta” ja siirrytään kohti keskustaa. Nämä tavat antavat hyvän kuvan siitä, missä asiat menevät pieleen, jos menevät. Kaikki polut testataan yksitellen ja ongelmakohdat saadaan haarukoita paremmin esille.

Toiminnallisessa integraatiotestauksessa liittymät jaetaan omiin lokeroihinsa järjestelmän toiminnallisuuksien perusteella. Tämä tapa noudattelee ylhäältä alas -mallia, joskin testattavat kokonaisuudet ovat pienempiä.

Big Bang -mallissa testataan kaikki integraatiot kerralla, eli koko ketju alusta loppuun. Tapa on erittäin yksinkertainen, mutta aiheuttaa ongelmia ongelmien paikantamisessa, mikäli jokin menee pieleen. Usein myös top down ja bottom up -mallit saattavat muuntua Big Bang -malliksi, vaikkei se olisi tarkoitus. (Hass 2008, 12-13)

Wormsin kannalta olennaisia integraatioita on kahden tyyppisiä. SAPin kanssa Worms kommunikoi yhteensä kolmen SOAP-pohjaisen liittymän kautta (1 kpl sisään, 2 kpl ulos) sekä ebMS (Electronic Business Messaging System)-kuljetuskehyksiin pohjautuvan sähköisen rajapinnan kautta urakoitsijoiden kanssa.

Kommunikointi urakoitsijoiden suuntaan tapahtuu ulkopuolisen välityspalveluntarjoajan kautta. Viestit tallennetaan FTP-palvelimelle, josta välityspalvelu noutaa ne ja välittää urakoitsijoille. Vastavuoroisesti välityspalvelu tallentaa saapuvat viestit FTP-palvelimelle, josta Worms lukee ne sisään tietokantaan. TIEKE-sanomien mukaiset UBL-tiedostot tulee paketoida kuljetuskehykseen, joka Vattenfallin tapauksessa noudattaa ebMS-standardia. ebMS on sanoman siirtoprotokollasta riippumaton tapa kuljettaa minkä tyyppisiä tiedostoja hyvänsä. Se rakentuu SOAP-protokollan päälle ja toimii erinomaisesti Vattenfallin käyttämän FTP-siirtotavan kanssa. ebMS-viestit rakentuvat useasta MIME(multipurpose internet mail extension)-osasta, joiden sisällä voidaan välittää useampia viestejä ja liitteitä. (Tieke ry. 2010, 42-43)

4.10 Suorituskykycase: VR:n lipunmyyntijärjestelmän uudistus 2011

VR uudisti junalippujen myyntijärjestelmänsä syyskuussa 2011. Uudistettu verkkokauppa kaatui käytännössä välittömästi eikä lippuja pystynyt ostamaan muuta kuin poikkeustapauksissa. Myös kaikki asemilla olevat lippuautomaatit kaatuivat ja olivat poissa käytöstä useita päiviä.

Ongelmat johtuivat kapasiteetin väärästä arvioinnista. Järjestelmän toimittaneet Accenture ja Tieto olivat toimittaneet VR:lle arvion kapasiteettitarpeesta ja VR oli määritellyt palvelinkapasiteetin sen mukaan. Käyttöönoton jälkeen kapasiteetti jouduttiin kolminkertaistamaan, mutta sekään ei vielä riittänyt (HS 20.9.2011). Kapasiteettiongelmien lisäksi syntyi muita teknisiä ongelmia.

VR:n tietohallintojohtaja Jukka-Pekka Suokko kertoi Aamulehden uutisessa, että käyttäjämäärät olivat käyttöönottohetkellä 20-kertaiset normaaliin kuormaan verrattuna. Accenturen ja Tiedon kapasiteettimääritys perustui aiempiin kävijäpiikkeihin mm. kevään 2010 tuhkapilven ajalta, jolloin lentoliikenne oli pysähtynyt (AL 20.9.2011).

Kapasiteettiongelmien seurauksena VR on joutunut antamaan matkustajien kulkea junissa maksutta, koska heille ei ole yksinkertaisesti pystytty myymään lippuja. Myös asemien lipunmyyntipisteet ovat tukkeutuneet ja järjestelmä hidastellut. Tilannetta pahentaa se, että kaksi kolmesta lipunmyyjästä on korvattu automaateilla vuosien varrella. Tappioita oli vielä tämän työn kirjoitushetkellä mahdotonta arvioida.

Herääkin kysymys siitä, miksei kapasiteettitarvetta osattu ennustaa paremmin? Kyseessä oli kuitenkin mittava uudistus, jonka seurauksena koko VR:n hinnoittelumalli muuttui ja voisi olettaa, että se herättää tavallista enemmän mielenkiintoa järjestelmää kohtaan.

Paitsi että VR muutti hinnoittelumallia, se myös muutti sivujensa rakennetta sulauttamalla puhtaan aikatauluhaun osaksi verkkokauppaa. Onko esimerkiksi tämä yksi syy siihen, miksi kapasiteetti on mitoitettu huonosti? Mikäli järjestelmätoimittaja ei ole arvioinut pelkkiä aikatauluja etsineiden asiakkaiden kokonaismäärää ja tämän massan siirtämistä verkkokaupan sisälle, voi nopeasti syntyä suuri ero aikaisempien ja nykyisten käyttäjämäärien suhteen. Samalla tästä saadaan myös käytettävyyspuolen ongelma, koska nyt käyttäjä ohjataan tavallaan turhaan verkkokauppaan, kun tämä haluaisi katsoa vain aikatauluja. Tämä voi hämmentää osaa käyttäjistä.

Accenturen edustaja kommentoi Helsingin Sanomille ongelmia toteamalla: ”Tämä on ollut monitoimittajahanke, jossa eri toimittajat ovat toimittaneen sekä uusia sovelluksia että muutoksia VR:n nykyisiin sovelluksiin”. Tämä herättää lisää kysymyksiä, esimerkiksi miten eri osapuolten välistä toimintaa on koordinoitu ja miten testaaminen on suoritettu, jotta kaikki integraatiot ja tekniikan yhteensovittaminen on saatu käytyä kunnollisesti läpi? Mikäli osapuolia on useampi, pitäisi ainakin silloin testien olla erityisen tarkkoja.

4.11 Käytettävyyscase: Potilastietojärjestelmä

Valtiontalouden tarkastusvirasto teki vuoden 2011 aikana tuloksellisuustarkastuksen, jossa tutkittiin sosiaali- ja terveydenhuollon valtakunnallisten IT-hankkeiden toteutuksia. Hankkeella pyrittiin mm. selvittämään eri organisaatioiden valmiuksia siirtyä käyttämään suunniteltua KanTa-potilastietojärjestelmää, jonka tarkoitus oli vastata potilastiedon lisääntymiseen ja sen siirtämiseen muihin järjestelmiin. Tietomäärän lisäksi haasteita on järjestelmien monimutkaisuudella ja lääkäreiltä kuluukin 43 prosenttia potilasta kohden varatusta työajasta tietokoneen käyttöön. (Valtion tarkastusvirasto 2011, 30)

Tutkimuksesta käy ilmi, että terveydenhuollon organisaatiot käyttävät yhteensä seitsemää erilaista potilaskertomusjärjestelmää (luvussa ei ole mukana hammashuoltoa ja työterveyttä), joiden lisäksi niitä tukemassa on useita liitännäissovelluksia, kuten laboratoriotesteihin liittyviä järjestelmiä. (Valtion tarkastusvirasto 2011, 31)

Eri järjestelmät eivät pysty aktiivisesti kommunikoimaan keskenään, vaan esimerkiksi laboratoriotulokset joudutaan siirtämään käsin järjestelmästä toiseen. Myöskään järjestelmässä suoritetavat työvaiheet eivät tue todellista työtä. Esimerkiksi potilaan tietoja voidaan selata ainoastaan yhdessä ikkunassa, vaikka tietoa löytyy useasta eri paikasta. Mikäli hoitava henkilö joutuu tarkistamaan potilaan hoitoon vaadittavia lisätietoja, hän joutuu sulkemaan auki olevan näkymänsä ja siirtymään toiseen osaan järjes-

telmää. Tämän jälkeen hän joutuu palaamaan takaisin alkuperäiselle ruudulle, jotta voi hyödyntää juuri löytämäänsä tietoa. (Valtion tarkastusvirasto 2011, 31-34)

Lääkärit kokevat eri järjestelmiin kirjautumisen hitaana, syrjäisempien toimipisteiden hitaammat internetyhteydet estävät röntgenkuvien tehokkaan tarkastelun ja mahdolliset katkokset tietoverkossa lukitsevat käsittelyssä olleen potilaan tiedot, kunnes järjestelmä aikakatkaisee istunnon. Näin ollen lyhytkin katko tietoliikenteessä voi hidastaa potilaan hoitoa merkittävästi. Selvityksessä on muodostunut käsitys, että ”järjestelmiä ei ole suunniteltu, testattu ja varmennettu riittävästi vaativaan terveydenhuoltoympäristöön sopivaksi”. (Valtion tarkastusvirasto 2011, 34)

Tietojen siirto eri organisaatioiden välillä ei toimi, joten 67 prosenttia Lääkärilehden kyselyyn vastanneista lääkäreistä ilmoitti edelleen pyytävänsä tiedot paperilla tai faksilla.

Oulun seudun yhteispäivystyksessä oleva laboratoriojärjestelmä kommunikoi kyllä potilastietojärjestelmän kanssa, mutta tiedon siirto järjestelmästä toiseen voi viedä 15-30 minuuttia, joka voi olla päivityspoliklinokoille merkittävä potilaan terveyden kannalta. Päivystyspoliklinikat joutuvat myös altavastaajan asemaan erilaisten päivityskatkojen takia, koska ne ajoittuvat ilta- ja viikonloppuaikaan, jolloin päivystyspoliklinikoilla on yleensä ruuhkaisinta.

HUSissa käyttöön otettu Miranda-potilastietojärjestelmä koettiin 2000-luvun alussa niin hankalaksi käyttää, että paperiset potilaskertomukset olivat ainoita ”virallisia” potilaskertomuksia. Tekstin syöttäminen potilaskertomukseen oli niin hankalaa, että lääkärit eivät tehneet sitä lainkaan, vaan sanelivat tekstin nauhurille ja antoivat sen konekirjoittajalle.

Tietokantarakenne saa myös kritiikkiä. Järjestelmät itsessään eivät tarjoa raportointityökaluja, jonka lisäksi tieto on järjestelmissä niin pirstaloitunutta, että sitä ei pystytä tehokkaasti raportoimaan erillisillä raportointityökaluilla. Peruskäyttäjän on mahdo-

tonta tuottaa omia raportteja. Uudet raportit joudutaan tilaamaan järjestelmätoimittajalta.

Potilastiedoissa henkilölle on voitu määritellä lääkeaineallergiat koodimuotoisina, mutta, mutta lääkemääräyksissä saatetaan silti sovelluksesta riippuen määrätä lääkkeet tekstimuotoisina, jolloin järjestelmä ei varoita mikäli potilas on kirjoitetulle lääkeaineelle allerginen. Datan säilytys eri muodoissa voi siis koitua potilaan kohtaloksi. (Valtion tarkastusvirasto 2011, 35-37)

Yhteenvetona voi siis todeta, että terveydenhuoltopalvelujen järjestelmiä rakennettaessa ei ole huomioitu lainkaan käyttäjiä ja sitä, miten prosessi oikeasti toimii. Toimialueella, jossa ajantasainen ja virheetön tieto olisi elinehto, on sen saaminen käyttäjille tehty monilta osin mahdottomaksi. Järjestelmien monimutkaisuudesta hyötyy ainoastaan ohjelmistotoimittaja, jolta joudutaan jatkuvasti tilaamaan korjauksia ja parannuksia.

Yllä kuvattu tilanne Suomen terveydenhuoltojärjestelmistä on melko karua luettavaa ja on vain pintaraapaisu miltei 300-sivuisesta tarkastuskertomuksesta, jota voi suositella luettavaksi kaikille asiasta kiinnostuneille. Sivuhuomiona esimerkiksi sähköisen potilas- ja reseptijärjestelmän kehittämiseen on toistaiseksi kulutettu laskutavasta riippuen n. 60-80milj. euroa, eikä järjestelmä ole vieläkään täysin käyttökuntoinen. (Valtion tarkastusvirasto 2011, 55-105).

5 MENETELMÄT

Projektin aikana käytetyt menetelmät jaetaan valmisteleviin ja suorittaviin menetelmiin. Jo järjestelmän suunnitteluvaiheessa tietoa kerättiin useasta lähteestä mahdollisimman kattavan vaatimusmäärittelyn tueksi. Järjestelmän rakentamisen aikana huomiota kiinnitettiin Avuxin ongelmiksi muodostuneisiin asioihin ja ratkaisuja testattiin käytännössä kehityksen loppupuolella. Opinnäytetyön osuus keskittyy enemmän toiminnalliseen puoleen.

5.1 Esiselvitys

Yrityksen sisäisille Avux-käyttäjille, joita on n. 100, tehtiin alkuvuodesta 2011 sähköinen kysely (liite 1), jossa kartoitettiin Avuxin puutteita ja vahvuuksia sekä asioita jotka tulisi ottaa huomioon järjestelmää päivitettäessä. Samalla kysyttiin mielipiteitä liittyen urakoitsijoiden omien tietojärjestelmien hyödyntämiseen ja raportoinnin kehittämiseen. Vapaan sanan kommenteilla pyrittiin kartoittamaan ikääntyneen järjestelmän sellaisia ominaisuuksia, joita ei välttämättä käytä kuin murto-osa henkilöstöstä ja jotka olisivat saattaneet jäädä huomioimatta muutosprojektin aikana.

Kysely oli erityisen tärkeä niin opinnäytetyön, kuin myös kehittämistehtävän kannalta. Koska kyseessä on melko mittava järjestelmäuudistus, voitiin kyselyn tuloksia käyttää asettamaan uudelle järjestelmälle tarvittavat laatuvaatimukset sekä rakentaa niiden pohjalta soveltuva testisuunnitelma. Lisäksi muutosvastarintaa on helpompi alkaa ehkäisemään jo etukäteen, kun henkilöstö otetaan mukaan projektiin jo sen alkuvaiheessa.

5.2 Vaatimusmäärittely ja asiantuntijalausunnot

Ohjelmistolle on laadittu erillinen vaatimusmäärittely, joka testien aikana käydään kohta kohdalta läpi kehityksen eri vaiheissa. Lista toimii myös Scrum-projektin product

backlogina ja testissä hyväksytyt ominaisuudet jäädytetään lopullista järjestelmäversiota varten.

Product backlogilla olevien ominaisuuksien testaamiseen käytetään organisaation eri yksiköiden asiantuntijoiden laatimia testitapauksia, joilla pyritään suorittamaan mahdollisimman laajalla skaalalla erilaisia tilaussanomia ja käyttötapauksia, joita uuden järjestelmän pitäisi pystyä käsittelemään.

Projektin aikana järjestelmätestaukseen ja projektiryhmän kokouksiin osallistui eri prosessien asiantuntijoita Vattenfall Verkon organisaatiosta. Tämän referenssiryhmän kanssa käytyjen keskustelujen pohjalta pystyttiin projektin aikana varmistumaan siitä, että järjestelmän toiminta tukee prosessien työtä ja että ylipäätään keskitytään oleellisiin asioihin. Käytännössä eri osa-alueisiin keskittyviä pieniä workshop -tyyppisiä kokouksia järjestettiin aina silloin, kun jokin tietty järjestelmän osa-alue oli tulossa ohjelmoitavaksi.

5.3 Koneellinen testaus

Testivaiheessa palvelimelle ei saada todellisia loppukäyttäjiä niin paljon, että tehokas kuormatestaus olisi mahdollista. Tätä varten raskasta käyttäjäkuormaa simuloidaan Apache JMeter-ohjelmalla. Ohjelma muodostaa halutunlaisia kyselyitä palvelimelle ja sillä voidaan esimerkiksi testata tietokannan toimintaa lähettämällä kerralla paljon kyselyitä. Ohjelma muodostaa jokaisesta testistä aineiston, josta käy ilmi esimerkiksi vasteaika. Lisäksi SOAP-liikennettä testataan ja simuloidaan SOAP UI-ohjelmistolla, jonka avulla testataan erilaisten väärin muotoiltujen viestien vaikutusta järjestelmän toimintaan.

5.4 Manuaalinen testaus

Kuormatestauksen lisäksi järjestelmää testataan kahden tyyppisillä käyttäjillä. Nämä ryhmät ovat projektiin osallistuneet ihmiset, jotka hallitsevat järjestelmän sekä täysin

ummikot, joilla ei ole aiempaa kokemusta järjestelmästä. Molemmat ryhmät pyrkivät käyttämään ohjelmistoa niin tehokkaasti kuin mahdollista ja raportoimaan mahdollisista puutteista ja väärästä toiminnasta.

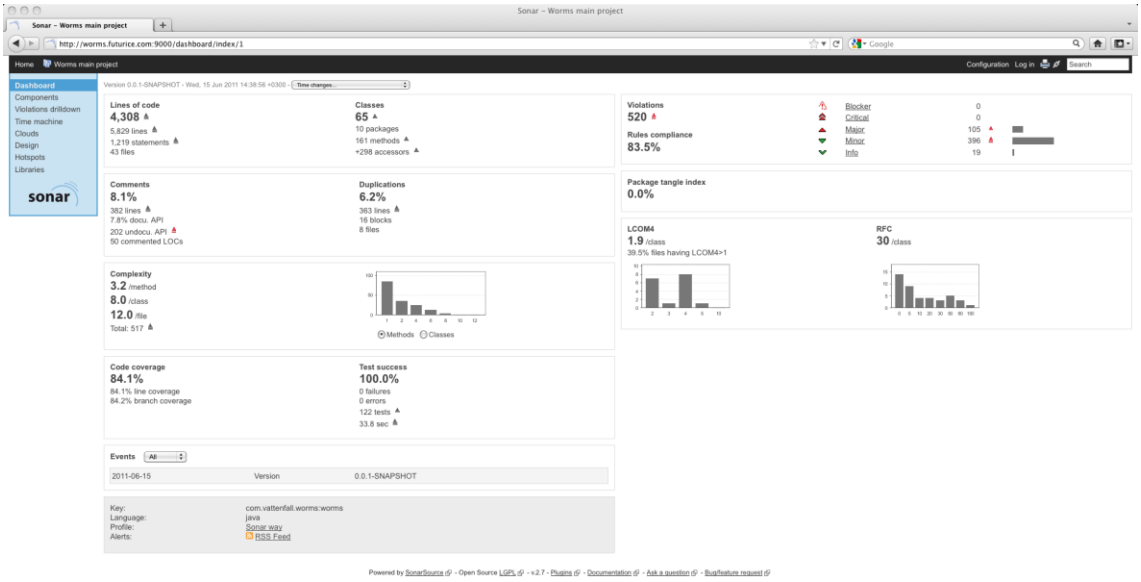
Järjestelmää yritetään myös väkisin käyttää väärin ja saada aikaan ei-toivottua toimintaa, joka voidaan korjata ennen kuin vastaava ongelma nousisi esiin ohjelmiston ollessa jo tuotantokäytössä.

6 JÄRJESTELMÄTESTAUS

Avux-järjestelmän heikkoutena on ollut sen sopimattomuus muuttuvaan toimintaympäristöön. Järjestelmän käyttöönottovaiheessa se oli rakennettu lyhytaikaiseen käyttöön ja optimoitu pienelle käyttäjämäärälle. Vuosien kuluessa sekä käyttäjä- että datamäärät ovat moninkertaistuneet.

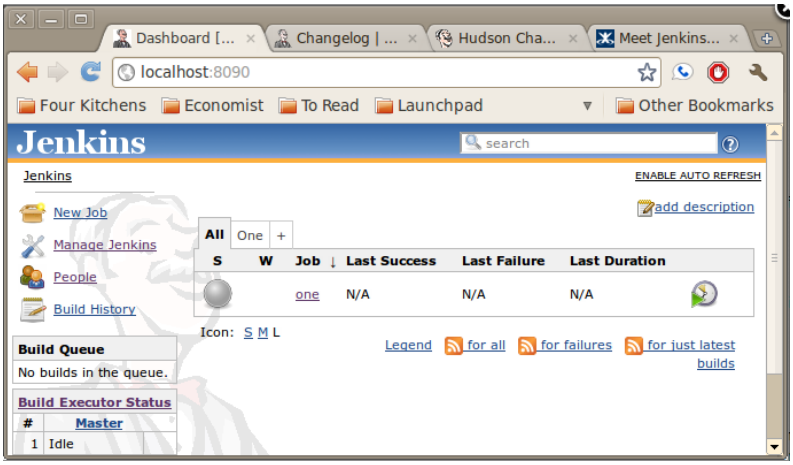
Jotta uuden Worms-järjestelmän kanssa ei päädyttäisi samaan tilanteeseen, tehdään sille kattava sarja erilaisia testejä pitäen silmällä myös mahdollisesti kasvavia vaatimuksia mm. suorituskyvyn ja jatkokehityksen osalta. Testit mittaavat paitsi suoritettavien toimintojen oikeellisuutta, myös palvelimen suorituskykyä sekä samanaikaisten käyttäjien että tietokantakyselyjen osalta. Myös järjestelmän käyttövarmuuteen ja huollettavuuteen paneudutaan.

Suurimpien ongelmien välttämiseksi järjestelmän koodia optimoitiin Sprinttien aikana käyttämällä apuna Sonar- ja Jenkins-työkaluja. Sonar (kuva 7) on tarkoitettu ohjelmoijan työkaluksi mm. tarkistamaan koodia turhan toiston ja väärän rakenteen osalta. Lisäksi se antaa listauksen mahdollisista koodin bugeista jo ennakkoon, jotta niihin voidaan puuttua ennen varsinaista testausta. Ohjelmisto tuottaa raportteja, joista selviää koodin erilaiset ongelmat kategorisoituna kriittisyyden perusteella. Tavoitteena on tuottaa 100 % testattua ja optimoitua koodia. Wormsin osalta testikattavuus oli sprintien päättyessä aina 100 %, joka on siis paras mahdollinen taso. Kuten esimerkkikuvasta käy ilmi, oli koodissa kuitenkin jonkin verran etenkin Minor-tason ongelmia, joita ratkottiin kehityksen loppupuolella.



Kuva 7: Sonarin dashboard, jossa Wormsin tunnuslukuja kehitysvaiheessa

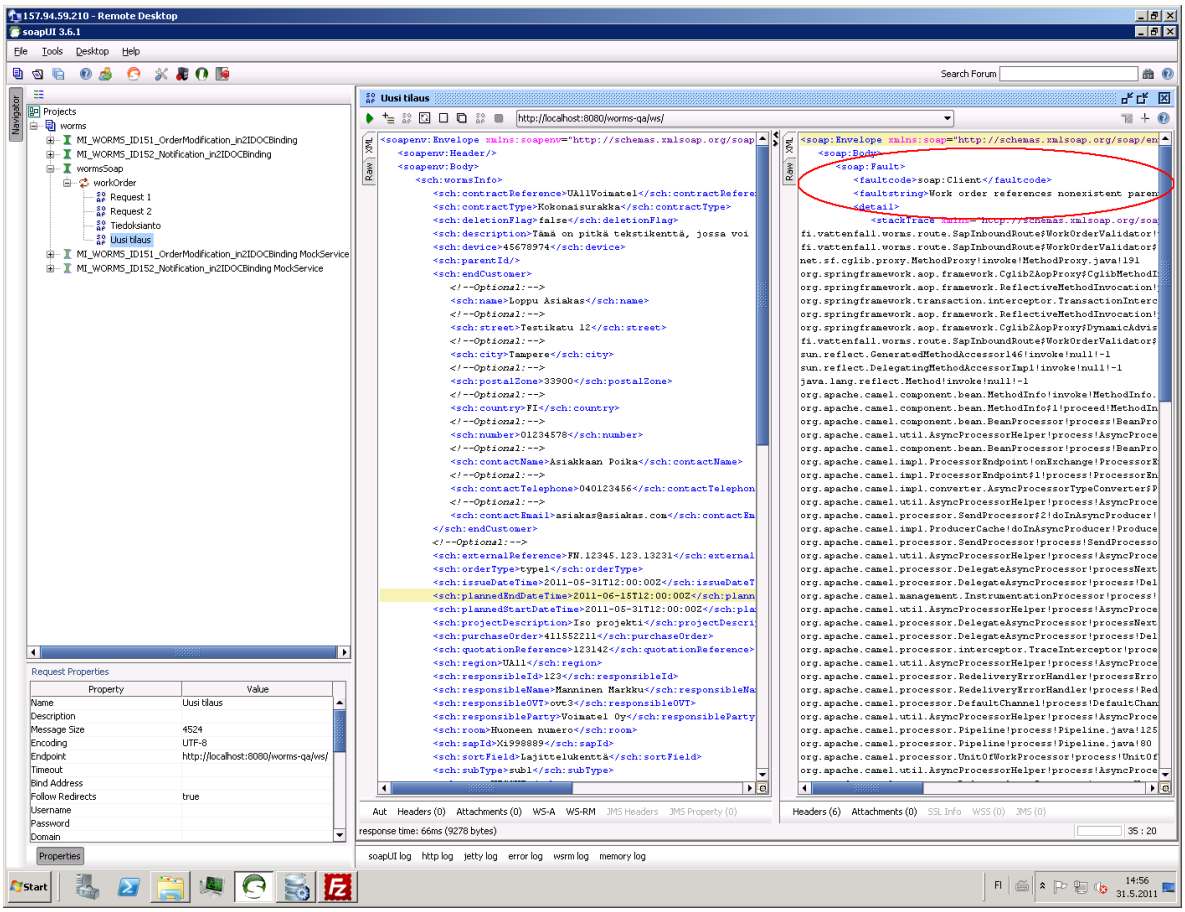
Jenkins (kuva 8) on puolestaan ohjelmistokehityksen tukena käytettävä versionhallintatyökalu, jonka avulla pystyttiin seuraamaan versioiden (build) käyttöönottoa ja analysoimaan virheitä koodin kääntämisvaiheessa. Ohjelmiston avulla voitiin nopeasti palauttaa aiempi versio järjestelmästä, mikäli tuore versio ei olisi toiminut oikein.



Kuva 8: Jenkinsin perusnäkymä (Lähde: www.jenkins-ci.org)

Ennen kuin varsinainen integraatio Wormsin ja SAPin välillä oli toimintakunnossa, pystyttiin integraatiorajapintoja testaamaan tehokkaasti avoimen lähdekoodin SOAP UI-ohjelmistolla (kuva 9), jolla voitiin simuloida SAPista saapuvia viestejä ja seurata niiden käyttäytymistä Wormsin tietokannassa. Lisäksi SOAP UI pystyi simuloimaan SAPin

SOAP-käsittelyä, eli lähettämään OK- ja virheviestejä riippuen Wormsista lähteneen sanoman sisällöstä.



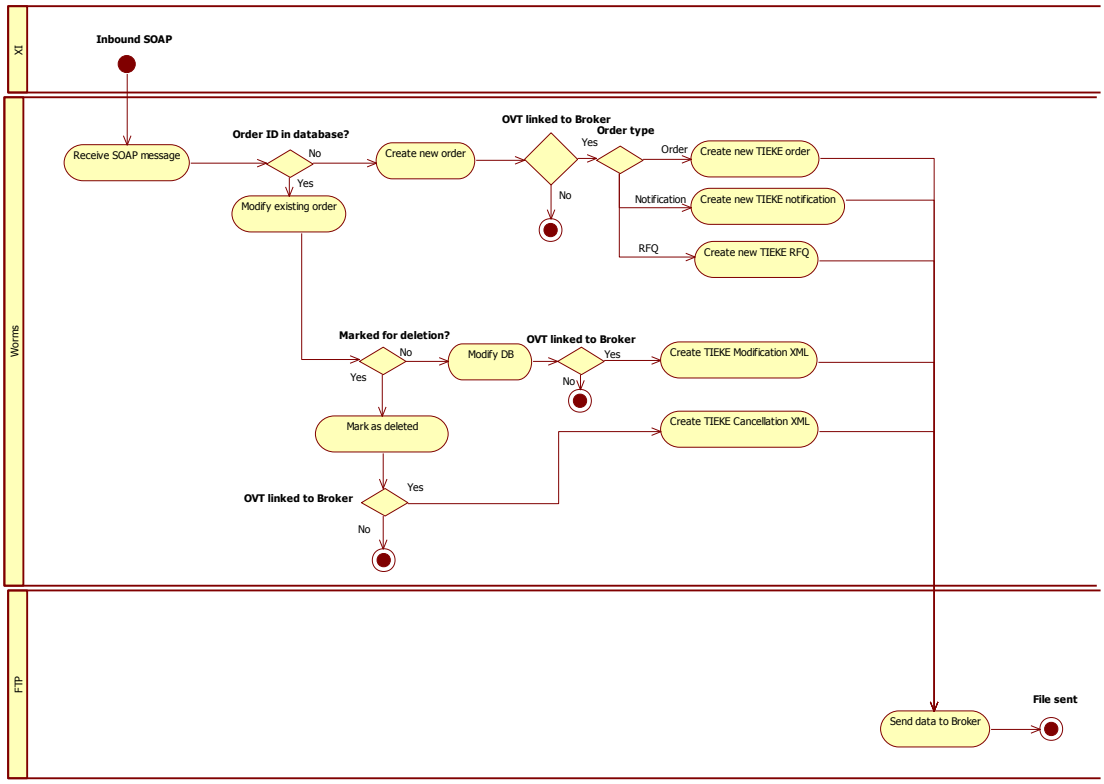
Kuva 9: SOAP UI:n käyttöliittymä, jossa näkyy Wormsin tuottama virhekoodi väärin muotoillusta SOAP-sanomasta.

Kuten myöhemmin tekstistä käy ilmi, SOAP UI oli korvaamaton työkalu selvitettäessä integraatio-ongelmia etenkin kehityksen alkuvaiheessa. Ohjelmistosta on saatavilla ilmainen perusversio, joka riitti tässä projektissa tehtyihin testeihin enemmän kuin hyvin.

6.1 Toiminnallisuus- ja integraatiotestaus

Toiminnallisuustestauksella käytiin läpi kaikki järjestelmän ominaisuudet, jotka vaatimusmäärittelyssä oli kuvattu. Olennaisimmat asiat olivat integraatiot Vattenfallin SAP-järjestelmään sekä urakoitsijoiden omiin ERP-järjestelmiin välityspalvelun kautta sekä

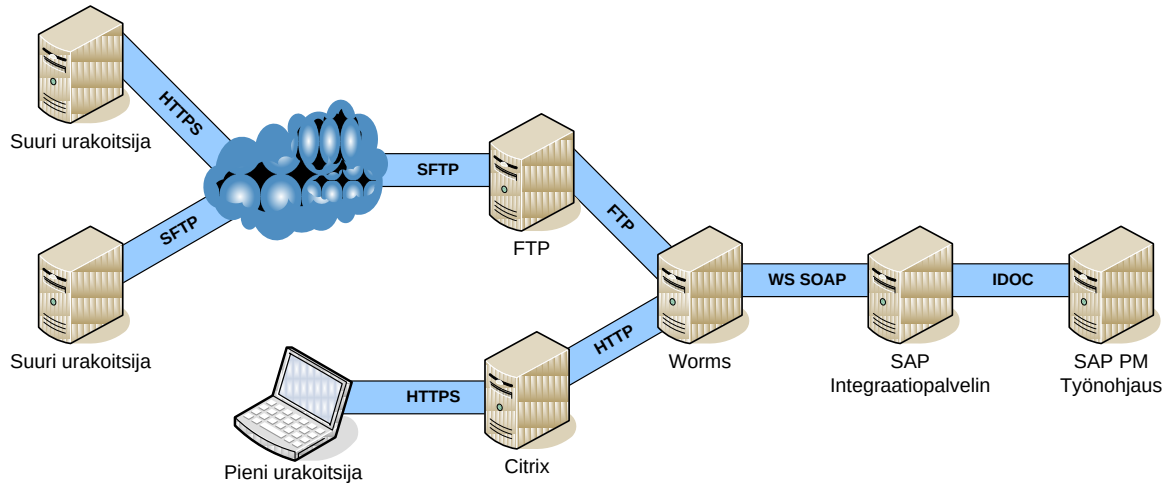
viestien käsittelyyn liittyvä automaatio Wormsin Apache Cameliin perustuvassa tilakoneessa, josta karkea esimerkki on nähtävissä kuvassa 10.



Kuva 10: Apache Camelin tilakonemäärittely tehtiin esimerkin kaltaisen UML-toimintokaavion avulla. Testit pohjautuivat myös siihen.

Tilakone tulkitsee SAPista saapuvaa SOAP-liikennettä ja ohjaa töiden tilaa sisällön mukaan. SAPista on ainoastaan yksi SOAP-liittymä Wormsin suuntaan, joka lähettää aina tilauksen kaikki tiedot riippumatta siitä onko kyseessä uusi työ, muutos vai peruutus. Mikäli tulkinnat olisi pitänyt tehdä SAPin puolella, olisi vaadittu huomattavasti monimutkaisempi koodirakenne ja todennäköisesti erillinen liittymä jokaiselle erityyppiselle tilausmuutokselle.

Kuvassa 11 esitetään erilaiset, Wormsin kannalta olennaiset integraatiot. SAPin päässä ensisijainen ”keskustelukumppani” on SAP XI -integraatiopalvelin, urakoitsijoiden suuntaan ainoastaan FTP-kansio. Web-käyttäjälle tieto esitetään Apache Tomcatissa ajettavalla Apache Wicket käyttöliittymällä, johon on puhtaisiin SQL-kyselyihin pohjautuvan tietokantaliittymän sijaan erillinen ns. RESTful (Representational State Transfer) Web Services -liittymä Wormsin tietokannasta.



Kuva 11: Wormsin kannalta olennaiset järjestelmäintegraatiot karkeasti kuvattuna.

Testeissä tutkittiin laajasti sitä, miten tieto kulkee järjestelmien välillä ja toteutuvatko Wormsin tilakoneeseen määritetyt rajoitukset ja työnkulkujen ohjaukset viestien sisällöstä riippuen.

Työnkulut vaihtelevat sen perusteella, toimiiko tilauksia vastaanottava urakoitsija omassa järjestelmässään vai käyttääkö hän Vattenfallin tarjoamaa web-palvelua. Kaiken perustana on kuitenkin tiedonsiirto Wormsin ja SAPin välillä SOAP- ja IDOC-muodossa.

6.1.1 SAP – Worms

Tärkeysjärjesteyksessä ensimmäisenä oli taata tilausviestien liikkuminen SAP-järjestelmästä Wormsiin.

Testeissä kiinnitettiin huomiota seuraaviin asioihin:

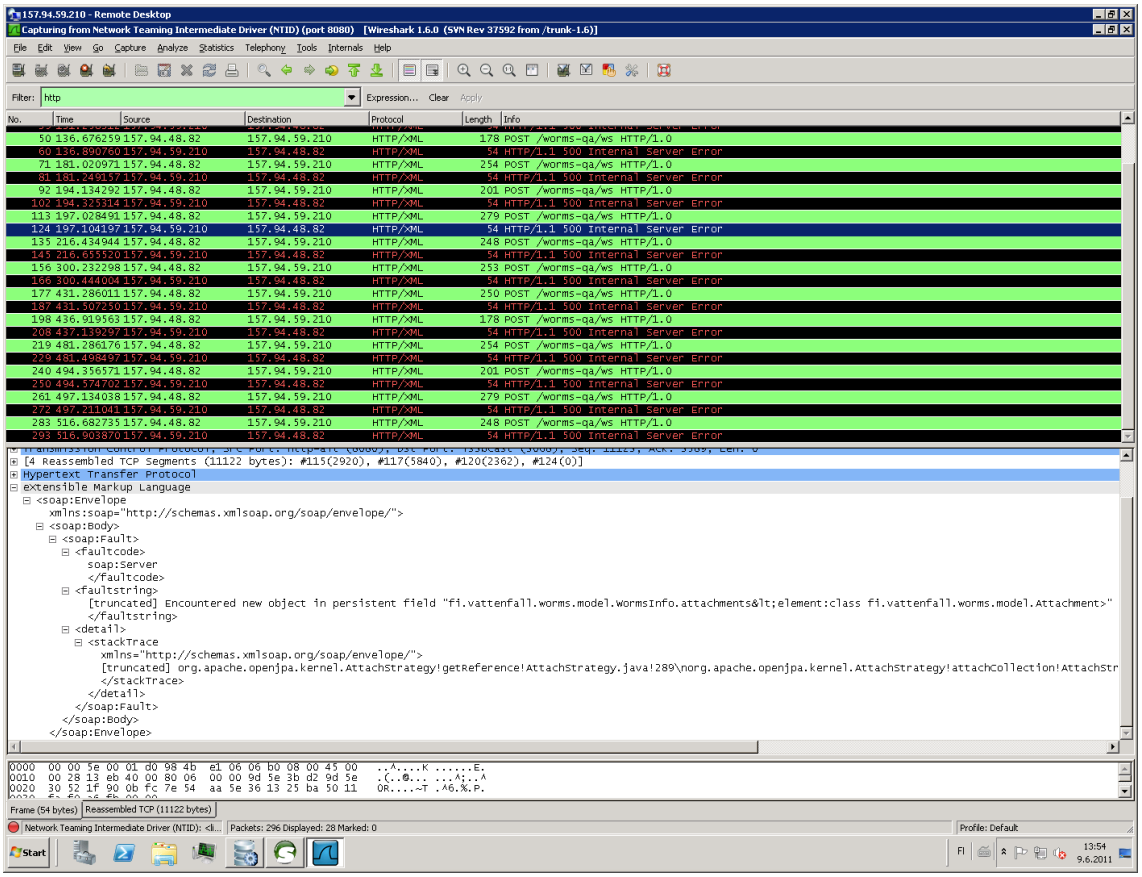
- Oikeiden tietojen välitys PM-moduulista XI:lle (IDOC)
- Oikein muotoillun viestin välitys XI:ltä Wormsiin (SOAP)
- Tietojen kirjautuminen Wormsin tietokantaan
- Työn tilan muuttuminen määrittelyjen mukaan

- Virheellisten sanomien käsittely

Käytännössä SAPin omat, todella tiukat rajoitukset ohjasivat järjestelmien välisen tiedonvaihdon hyvin standardiksi ja normaalissa käytössä voidaan olla varmoja siitä, että kaikki pakolliset tiedot siirtyvät oikeassa muodossa järjestelmästä toiseen. Varmuuden vuoksi tehtiin SOAP UIta apuna käyttäen lukuisia simulaatioita tilanteista, joissa siirrettävässä datassa viljeltiin erikoismerkkejä, välilyöntejä, rivinvaihtoja ja näiden yhdistelmiä. Tällä varmistuttiin siitä, että Worms hallitsee saapuvassa viestissä olevat erikoistilanteet ja käsittelee viestejä oikein sekä että Worms osaa muuntaa myös urakoitsijan suunnalta saapuvan virheellisen datan SAPin vaatimaan muotoon, joko poistamalla erikoismerkkejä tai rajoittamalla siirrettävien kenttien sisältöä muulla tavalla.

Testien alkuvaiheessa, kun SAP-Worms -väliä testattiin virallisesti ensimmäistä kertaa, oli ongelmia saada data siirtymään ns. end-to-end -välillä kulkemaan aina ISUn työtilaukselta TIEKE XML:ksi. Simuloitu SOAPin lähetys XI-palvelimelta toimi hyvin, mutta kun tietoa yritettiin siirtää ensin IDOC muodossa ISUsta XI:lle, ei kaikki toiminutkaan niin kuin piti.

Ongelmaa lähdettiin paikantamaan tutkimalla sitä, minkälainen SOAP-viesti XI:ltä muodostuu, kun tilaus lähetetään IDOC-muodossa SAPista. SAPin muodostama SOAP-virhe ei kuitenkaan ollut samanlainen, kuin Wormsiin määritellyt virheet. XI-palvelin muunsi vastaanottamansa virheviestit geneerisiksi ”yleisvirheiksi” yksilöimättä mikä tieto viesteissä oli virheellinen. Ongelman ratkaisemista varten Wormsin palvelimelle asennettiin avoimen lähdekoodin Wireshark -pakettianalysaattori, jolla voidaan seurata palvelimen tietoliikennettä (kuva 12) ja porautua myös SOAP-viestien rakenteeseen.



Kuva 12: Wiresharkin käyttöliittymä, HTTP-pyyntöjen suodatus kytkettynä porttiin 8080

Kun Wireshark oli käynnissä, lähetettiin ISUsta uusi IDOC ja seurattiin sen kulkua Wormsiin. Viesti muuntui todellisuudessa aivan normaalisti SOAP-määrittelyn mukaiseksi XI-palvelimella ja se kulkeutui normaalisti Wormsiin. Käsittely päättyi kuitenkin virheeseen. Kuvassa 12 SAPista saapuvat SOAP-viestit päätyvät HTTP500 -virheeseen, josta selviää Wormsin muodostama yksilöity virheilmoitus.

Virhettä tulkitsemalla ongelma jäljitettiin Wormsin testipalvelimella olevaan SOAPin käsittelyongelmaan. Esimerkkitapauksessa Worms vaati, että viestissä olisi ollut tietoja työlle kuuluvista liitteistä, mikäli <Attachments> -elementti esiintyy viestissä. SAPin lähettämä viestirakenne sisältää aina kaikki SOAP-viestin elementit (SOAP UIlla simuloidussa XI-viestissä liite-elementit puuttuivat testien alkuvaiheessa), vaikka ne olisivat tyhjiä, joten viestinkäsittelylogiikkaa muutettiin Wormsin osalta hyväksymään myös tyhjiä kenttiä.

Wiresharkilla saatiin haarukoitua esiin muitakin viestin rakenteeseen liittyviä ongelmia. Yleisesti ottaen virheet olivat pieniä, mutta kiusallisia löytää. Ne liittyivät usein esimerkiksi SOAPin elementtien merkkikokoon (case sensitivity), eli esimerkiksi ID -> Id. Käsittelylogiikan muutosten tekeminen Wormsin koodiin oli kuitenkin melko yksinkertaista.

Kun SOAP-rakenne oli saatu kuntoon, huomio siirtyi saapuneen tiedon käsittelyyn Wormsin tietokannassa. Ensimmäisissä testeissä kävi ilmi, että dataa hukkui johonkin matkalla, eli se ei kirjautunut tietokannan kenttiin oikein. Viat liittyivät Javan ”getter-setter” -määrityksiin, jotka yrittivät kirjata tietoa väärässä muodossa tietokantaan.

Ongelmien korjausta odotellessa oli syytä pureutua siihen, mitä getter-setter -toiminnallisuudet todella ovat. Selvisi, että yleisesti ottaen ei kyseistä toiminnallisuutta nykypäivänä suositella käytettäväksi oliopohjaisissa järjestelmissä (Holub 2003). Syynä on se, että niiden käyttäminen aiheuttaa paljon ylimääräistä koodausta ja sekavuutta, mikäli kohteena olevien kenttien (esim. Wormsin tapauksessa ”DateTime” -tyyppiset päivämäärät) tyyppiä jouduttaisiin tulevaisuudessa muuttamaan muuksi. Tämä herätti hieman huolta, sillä järjestelmän tarkoitushan oli olla mahdollisimman muuntautumiskykyinen, joten asia käsiteltiin järjestelmätoimittajan kanssa.

Metodien käyttö oli ohjelmoijien mielestä perusteltua, sillä Worms on enemmän ”perinteiseen” Javaan nojaava järjestelmä (ns. legacy-versio), jossa getterit ja setterit ovat vakiotoiminnallisuuksia. Uudet Java-variaatiot - joita hyödynnetään esimerkiksi Googlen Android-käyttöjärjestelmässä (joka ei ole Javaa, mutta käytännössä identtistä) - pohjautuvat kokonaisuutena erilaiseen toimintamalliin, jossa getter-setter -logiikka ei toimi. Wormsin koodin rakenne kokonaisuutena on sen verran yksinkertainen, että data pysyy helposti hallittavana. Koodin nopea muunneltavuus tietokantakenttien käsittelyssä todistettiinkin ohjelmoijien toimesta testien aikana useaan kertaan, joten huolenaihe oli turha. Asia on syytä pitää kuitenkin mielessä, mikäli järjestelmää tulevaisuudessa aiotaan laajentaa kuten Avuxia aikoinaan. Tällöin tulee arvioida kuinka paljon muutoksia ja jatkokehitystä voidaan sallia ilman että asiat monimutkaistuvat liiaksi.

Ongelmia esiintyi tiedon kohdistamisen lisäksi myös merkistöjen kanssa. Worms toimii oletusarvoisesti yleismaailmallista UTF-8 -merkistöä käyttäen, mutta yleisesti käytössä oleva ISO-8859-1 (”Western Latin”) -merkistö nousi usein esille ongelmana mm. tietokannan rakenteessa. Worms on tarkka siitä, että data liikkuu jatkuvasti UTF-8 -muodossa, joten sitä tulee painottaa etenkin urakoitsijoille välitettävässä dokumentaatiossa.

Kun onnistunut perusdatan siirto SAPista Wormsiin oli testattu, piti myös varmistua siitä että määritetyt muutos- ja peruutusviestit toimivat halutulla tavalla. Teknisesti SAPin päässä nämä viestit eivät eroa mitenkään uudesta tilauksesta (pl. peruutusviestin ”poistomerkki”), mutta Wormsin tilakone tulkitsee viestien sisältöä monipuolisesti. Mikäli työ löytyy jo tietokannasta ja sama ID saapuu uudestaan SAPin suunnasta, käsitellään työtä muutoksena. Tämän jälkeen tulkitaan mahdollista poistomerkkiä ja vasta sen jälkeen suoritetaan muuttuneiden kenttien päivitykset.

6.1.2 SAP – Worms – Välityspalvelu

Testiskenaariot, jotka liittyvät viestien välittämiseen urakoitsijoiden omiin järjestelmiin, laajentavat kohdan 6.1.1 skenaarioita siten, että Wormsin tilakone jatkaa viestien tietokantaan kirjaamisen jälkeen niiden prosessointia TIEKE-sanomaksi.

Riippuen SAPista tulevan datan sisällöstä, tilakoneen tulee osata muodostaa oikean tyyppinen UBL-sanoma. Tämä ei yksin riitä, vaan sanoma tulee ”paketoida” ebMS-kuljetuskehykseen, joka sisältää lähettäjän ja vastaanottajan tiedot. Täydellinen ebMS-sanoma pitää siirtää FTP-palvelimelle odottamaan välityspalvelun noutoajoa.

Avainasioita testauksessa olivat viestien tyyppien ja muuttuvan tietosisällön oikeellisuus sekä tiedon siirto FTP:lle. Viestin oikeellisuus tarkistettiin yhteistyössä Pohjolan Werkonrakennus Oy:n kanssa, joka vastaanotti sanomat ja tarkisti että ne kirjautuvat heidän järjestelmäänsä TIEKEn määritysten mukaan.

Lähtevien viestien osalta testattiin seuraavat TIEKE-sanomat:

- tilaus
- tilauksen muutos
- tilauksen peruutus
- tiedoksianto
- tarjouspyyntö
- tekninen kuittaus

Erityishuomiota kiinnitettiin työn tilaan tietokannassa. Urakoitsijoiden kanssa tehdyn määrittelyn mukaan muutos- ja peruutusviestejä ei voida enää toimittaa siinä vaiheessa, kun urakoitsija on aloittanut työt kohteessa (kun Worms on vastaanottanut ”työn alla” -sanoman). Graafisen käyttöliittymän työkäsittelyn lisäksi Wormsin tilakoneen on tehtävä ylimääräinen tulkinta tilasta, kun urakoitsija toimii sähköisessä rajapinnassa.

Avux käyttää sähköisessä rajapinnassa omaa tulkintaansa ebMS 3.0-standardista, mutta sen sanomien tulkinta perustui sanomien tiedostonimiin. Välityspalvelusta saapuvat viestit nimettiin palveluntarjoajan toimesta vastaamaan tiettyä sanomatyyppiä, jolloin Avux tiesi miten mitäkin sanomaa kuuluu käsitellä. Tämä on raskas tapa toimia, sillä jokainen urakoitsija vaati oman tietoyhteyden jokaista sanomatyyppiä varten, jonka lisäksi saapuvat viestit piti käyttää maksullisen sanomamuuntimen kautta palveluntarjoajan palvelimella.

Worms käyttää tiukemmin määriteltyä ebMS 3.0-standardia ja tulkitsee viestien tyypin puhtaasti sanomasisällön perusteella. Tämä tarkoittaa, että tietoyhteydet voitiin yksinkertaistaa radikaalisti. Palveluntarjoajan ei tarvitse nimetä sanomia uudelleen sisällön mukaan, vaan viestit välitetään sellaisinaan Vattenfallille.

Tietoyhteyksiä vaaditaan jatkossa vain yksi per urakoitsijayritys, koska kaikki olennainen tieto sanoman ohjaamiseen yritykseltä A yritykselle B löytyy ebMS:n otsikkotiedoista (lähettäjän ja vastaanottajan OVT-tunnukset). Tämä tuo mukanaan kustannussäästöjä, sillä palveluntarjoajien yleinen laskutusperuste on tietoyhteyksien lukumäärä. Myös uuden toimijan ottaminen mukaan sähköiseen rajapintaan aiheuttaa yleensä kertaveloituksen jokaisesta avattavasta tietoyhteydestä. Koska TIEKE-sanomia otetaan

kokonaisuudessaan käyttöön 12 kpl ja Avuxissa sanomia on 5, olisi kustannusten nousu ollut vanhalla tavalla toteutettuna merkittävä.

Samalla kun tietoyhteyksien määrää pystyttiin vähentämää, päivitettiin myös FTP-yhteys Vattenfallin ja palveluntarjoajan välille käyttämään salattua SFTP-protokollaa. Aiemminkin yhteys oli salattu, koska se muodostettiin VPN-yhteyden avulla, mutta VPN-palvelun kanssa on ollut usein ongelmia. Palveluntarjoajan kokemusten mukaan SFTP toimii tavallista FTP:tä luotettavammin ja sen käyttöönotto mahdollisti samalla VPN-yhteydestä luopumisen, mikä helpottaa ylläpitoa ja lisää entisestään toiminta-varmuutta.

Viestit muodostuivat pääsääntöisesti oikein FTP:lle. Ne sisälsivät ainoastaan pieniä puutteita XML-elementtien attribuuteissa, mutta teknisesti viestit olisivat toimineet varmasti jo tuotantoympäristössäkin oikein. Tämä johtuu siitä, että Wormsin rakentamat sanomat on mallinnettu koneellisesti XML-skeemojen pohjalta ja ovat siten identtiset UBL 2.0 standardin kanssa. Viestin manuaalisesta laatimisesta johtuvat rakennevirheet eivät siis muodostuneet ongelmaksi.

Jos vastaanottajan viestikäsittely olisi ollut äärimmäisen tarkkaa myös attribuuttien osalta, olisivat viestit kuitenkin jääneet virheeseen, joten kaikki havaitut puutteet korjattiin ennen varsinaisten testien aloittamista urakoitsijan järjestelmän kanssa.

Jos työ on jo aloitettu ja sille tehdään muutos SAPissa, Wormsin tulee hyväksyä muutos tietokantaan, muttei välittää sitä eteenpäin. Tämän tyyppisessä tapauksessa työn kuvasta päivitetään ylimääräisellä kommentilla, jossa viitataan myöhäiseen muutos-ajankohtaan. Näin ollen Vattenfallin käyttäjät voivat myöhemmin nähdä järjestelmästä miksi muutos ei urakoitsijan edustajaa tavoittanut. Käyttäjäystävällisempi ratkaisu olisi ollut rakentaa SAPin päähän rajoitukset, jolloin työlle ei voida kirjata muutoksia lainkaan kun tietyt ehdot täyttyvät, mutta ongelmaksi olisi muodostunut sisäisille toimijoille kohdistettujen töiden käsittely. Yli puolet tilauskäsittelystä liittyy Vattenfallin sisäisten käyttäjien väliseen toimintaan ja työn tietoja on niissä tapauksissa voitava muuttaa milloin vain.

6.1.3 Worms – SAP

Kun urakoitsija on vastaanottanut tilauksen, tulee työn etenemisestä raportoida Vattenfallin suuntaan (milloin työ on aloitettu, milloin valmistunut jne.). Tätä varten SAPin ja Wormsin välillä toimii kaksisuuntainen tiedonvaihto, jolla päivitetään järjestelmien tietoja.

Kun urakoitsija ottaa työn käsittelyyn, aloittaa työt kohteessa tai kuittaa työn valmiiksi, tulee siitä kirjata tieto Wormsin tietokantaan sekä välittää tiedot SAPiin. Erityyppiset tilannepäivitykset vaihtavat työn tilaa Wormsin tietokannassa ja aktivoivat ”tilausmuutos”-liittymän SAPin suuntaan. Olemassa olevalle tilaukselle päivitetään päivämääräarvoja sekä toteutuneita työyksiköitä.

Urakoitsijan suunnasta vastaanotetaan myös uusia tiedoksiantosanomia sekä tarjouksia. Nämä käyttötapaukset luovat Wormsin tietokantaan uuden rivin ja aktivoivat ”uusi tilaus”-liittymän SAPin suuntaan. Liittymä poikkeaa tilausmuutoksesta siten, että se aktivoi SAPin perustamaan uuden työn automaattisesti sen sijaan, että se päivittäisi olemassa olevan työn arvoja.

Nämä kaksi liittymää olivat melko haasteellisia SAPin työnkulkukäsittelyn johdosta ja vaativat useita korjauksia toimiakseen oikein. Vaikka Worms välittää kaikki olennaiset tiedot yhtenä SOAP-sanomana, joudutaan eri arvot purkamaan useaan paikkaan SAPissa, jotta ne kirjautuvat oikein varsinaiseen työnohjausnäkymään.

Urakoitsijan luomien tiedoksiantojen ja tarjousten mukana kulkee myös liitteitä. Avuxin kanssa liitteet on joko lisätty SAPissa tai ne ovat olleet saatavilla ainoastaan Avuxissa. Wormsin tapauksessa liitteitä tuli siirtää SAPiin myös järjestelmän ulkopuolelta. Worms siis tallentaa saapuvat liitteet omalle levylleen ja välittää SAPiin tiedon tiedostojen sijainnista, tiedostotyypistä ja -koosta. SAP piti määritellä lukemaan tämä tieto sisään ja luomaan omaan tietokantaansa viitteet oikeaan hakemistoon. Useiden testien jälkeen ominaisuus saatiin toimimaan.

6.1.4 Välityspalvelu – Worms – SAP

Edellisen alakappaleen (6.1.3) toiminnallisuuksia täydennetään sähköisessä rajapinnassa toimivien urakoitsijoiden osalta. Tämä tarkoittaa, että Wormsin on kyettävä käsittelemään FTP-palvelimelle saapuvia ebMS-sanomia hyvin monipuolisesti.

Saapuvia sanomia ovat tekniset kuittaukset, tilauskuittaukset, työn aloitus- ja lopetusilmoitukset, toimitusluettelot, tiedoksiannot sekä tarjoukset. Wormsin tulee saapuvan sanoman sisällöstä päätellä minkä tyyppinen viesti on kyseessä, mihin työhön se liittyy ja mitä sille pitää tehdä. Osa viesteistä päivittää tietokannassa jo olevia töitä, osa perustaa uusia töitä.

Tietojen tarkistus sisäänlukuvaiheessa on olennaista. Vain tiettyjä tietokannan arvoja on mahdollista muuttaa urakoitsijan suunnasta tulevilla viesteillä. Virhekäsittelyn testaaminen kuului myös kokonaisuuteen. Mikäli urakoitsijalta vastaanotettiin väärin muodostettu tai sisällöltään muuten kyseenalainen sanoma, tuli se hylätä ja lähettää ilmoitus urakoitsijalle.

Saapuvista TIEKE-sanomista testattiin seuraavat:

- tilauksen vastaanottokuittaus
- tekninen kuittaus
- työn alla -ilmoitus
- työ valmis -ilmoitus
- toimitusluettelo
- tarjous
- tiedoksianto

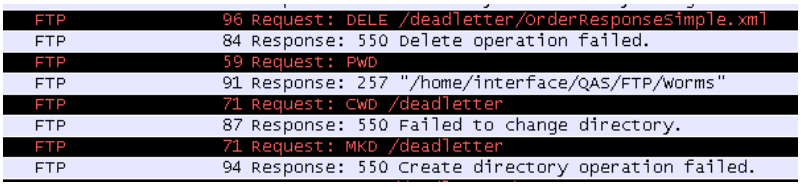
Testauksen alkuvaiheessa esiin nousi hyvinkin olennaisia puutteita, jotka olisivat ilman toimenpiteitä voineet aiheuttaa paljon vahinkoa. Esimerkiksi urakoitsija pystyi tilauskuittauksella muuttamaan työn vastuu-urakoitsijaa – eli siirtämään työn toisen yrityksen nimiin – manipuloimalla kuittauksen lähettäjän tunnistetietoja.

Havaitut ongelmat saapuvien viestien tietojen käsittelyssä korjattiin tekemällä tiukat rajoitukset sille, mitä kenttiä tietokantaan voidaan lukea sisään. Vain tiettyjen kenttien

arvoihin sallitaan siis muutoksia. Urakotisijan on lisäksi toimitettava olennaiset sanomat oikeassa järjestyksessä ja oikeilla tiedoilla, jotta työn tila päivittyy järjestelmässä normaalisti. Käytännössä työtilaukselle tulee saada urakoitsijan suunnalta vastaanottokuittaus, aloituskuittaus, lopetuskuittaus ja toimitusluettelo, jonka jälkeen se on täydellisesti valmistunut.

Ongelmia oli myös FTP-palvelimen kanssa. Vaatimusten mukaan Wormsin tulee siirtää virheelliset sanomat odottamaan tarkempaa tutkimusta omaan kansioon FTP:llä. Virheestä lähti myös tieto sanoman lähettäjälle sähköpostilla. Jostain syystä virheviestit eivät kuitenkaan poistuneet saapuvien viestien kansiosta, vaan jäivät ikuisesti pyörimään sinne, aiheuttaen loputtoman määrän virheilmoituksia lähettäjälle.

Esimerkkitapausta ei saatu toistettua kehitysympäristössä, joten sitä lähdettiin tutkimaan Wiresharkin avulla, jolla seurattiin Wormsin ja FTP-palvelimen välistä viestintää. Tutkimuksissa selvisi, että Wormsin FTP-toiminnallisuus ei noudattanut asetettua käsittelyjärjestystä, vaan yritti suorittaa komentoja melko sattumanvaraisessa järjestyksessä. Missään vaiheessa Worms ei lähettänyt FTP:lle siirtokomentoa, vaan suoritti muita toimintoja uudestaan ja uudestaan. Järjestelmä yritti mm. poistaa virheellistä viestiä virhekäsittelykansiosta, ennen kuin oli suorittanut siirtokomentoa. Myös virhetiedostokansion käsittelyssä oli ongelmia (kuva 13).



Kuva 13: Wiresharkin tuottamaa lokitietoa Wormsin FTP-käsittelyongelmasta.

Vika johtui lopulta siitä, että kehitysympäristö pystyi toimimaan relatiivisten hakemistopolkujen avulla, mutta Vattenfallin testiympäristö vaati absoluuttiset hakemistopolut. Tämä esimerkki on oiva osoitus siitä, että ohjelmistokehitys on ongelmallista silloin, kun kehitys tapahtuu jossakin muussa kuin lopullisen tuotantoversion kanssa identtisessä ympäristössä.

Kun FTP-liittymä oli saatu toimimaan, aloitettiin virallinen testaus. Testit jaettiin kolmeen osaan, joista yhdessä käsiteltiin urakoitsijan vastaanottama työtilaus, toisessa vastattiin Vattenfallin lähettämään tarjouspyyntöön tarjouksella ja kolmannessa pyydettiin työlle lisätietoja tiedoksiannon muodossa.

Worms ei validoi TIEKE-sanomien sisältöä niin tarkasti kuin voisi vaatia. Se poimii ainoastaan työn etenemisen kannalta oleelliset tiedot sisällöstä ja hylkää muun tiedon kokonaan. Oleelliseksi tiedoksi ei esimerkiksi lueta vapaata tekstiä sisältävien kenttien kieliattribuutteja tai erilaisten standardointiorganisaatioiden tunnistetietoja, koska saapuvaa dataa peilataan tietokannassa olevaa tietoa vasten ja varsinaisen sisällön oikeellisuus ratkaisee.

Vastapainoksi järjestelmä on äärimmäisen tarkka sanomarakenteesta (XML namespacet) ja sanoman merkistöstä (puhdas UTF-8). Kun viestit olivat määritellyssä muodossa, ei niiden käsittelyssä syntynyt ongelmia.

6.1.5 Urakoitsija- ja käyttäjähallinta

Jotta SAPissa luotavat tilaukset kirjautuisivat Wormsiin, tulee järjestelmän tietokannassa olla OVT-tunnukseltaan SAPista saapuvaa sanomaa täsmäävä urakoitsija. Urakoitsijalle ei tarvitse määritellä ainuttakaan käyttäjää, mikäli tämä toimii sähköisessä rajapinnassa. Urakoitsijoita hallitaan ainoastaan tietokantaa päivittämällä eikä erillistä hallintatyökalua rakennettu tässä vaiheessa. Käytännössä urakoitsija koostuu OVT-numerosta, hallinnollisesta sähköpostista ja ”rajapintakytkimestä” (0 tai 1).

Kevyttä web-käyttöliittymää varten luotiin ylläpitotyökalu, jolla voidaan hallita käyttäjiä. Käytännössä työkalu on ainoastaan graafinen käyttöliittymä Wormsin tietokantaan, jotta henkilöt, joilla ei ole tietokantaosaamista voivat ylläpitää käyttäjätietoja ilman että arvoja tarvitsee päivittää suoraan tietokantaan. Muutamalle Vattenfallin avainkäyttäjälle on määritelty käyttöliittymään ns. Admin-tunnukset, joilla päästään käsiksi käyttäjätietoihin.

Käyttäjien lisääminen liittyy läheisesti kohdassa 6.2.1 läpi käytävään käyttöliittymätestiin. Yhdellä käyttäjällä voi olla aktivoituna ainoastaan yksi urakoitsijayritys. Käyttäjä voi tuolloin nähdä web-käyttöliittymässä ainoastaan ko. urakoitsijalle kohdistettuja töitä.

Työkalun toiminnallisuutta testattiin luomalla uusia urakoitsijoita tietokantaan sekä lisäämällä osalle niistä käyttäjiä web-työkalulla. Asetusten asettamista ja muuttamista testattiin monella eri variaatiolla. Käyttäjien salasanat tallennetaan kantaan kryptattuina ja SALT-salausalgoritmilla varustettuna.

Työkalu on melko yksinkertainen ja toimi alusta asti toivotulla tavalla. Työkalulla kirjatut käyttäjätiedot tallentuivat tietokantaan oikein ja uusi käyttäjä pystyi kirjautumaan käyttöliittymään.

6.1.6 Raporttien koostaminen

Worms toimii Vattenfallin pääasiallisena raportointialustana työnohjaukseen liittyvän datan osalta. Tätä varten järjestelmän toiminnallisuutta testattiin erilaisilla raportointityökaluilla. Tarkoituksena oli varmistaa että tietokantaan saadaan yhteys Wormsin ulkopuolelta ja että kaikki tietokannan taulut ovat käytettävissä.

Koska Wormsin tietokanta on MySQL-pohjainen, pystytään siihen kytkeytymään valtaosalla markkinoilta löytyvistä raporttityökaluista joko suoraan tai hyödyntämällä ODBC-ajureita, jotka ovat vapaasti saatavilla. Testeihin käytettiin RazorSQL-, Microsoft Access-, SAP Crystal Reports-, sekä Qlikview-ohjelmistoja.

Testejä varten työasemalle asennettiin MySQL Connector 5.1.8 ODBC-ajuri sekä lisäksi RazorSQL:ää varten MySQL Connector 5.1.6 Java-ajuri.

RazorSQL ei varsinaisesti ole raportointi- vaan tietokantatyökalu, mutta se otettiin mukaan siksi, että tietokannasta voi olla tarve hakea joskus nopeasti tietoa ja tallentaa se

esimerkiksi CSV-muotoon. Tähän tarkoitukseen työkalu soveltuu paremmin kuin varsinaiset raportointiohjelmistot.

Eri ohjelmistot saivat vaivatta yhteyden tietokantaan, mikä on iso edistysaskel verrattuna Avuxin hivenen epästandardiin Firebird SQL-tietokantaratkaisuun, jolle ei ollut saatavilla kunnon ajuritukea. Firebirdin ODBC-ajuri on sekava ja sen avulla otetaan yhteys suoraan tietokantatiedostoon palvelimen kiintolevyllä sen sijaan että yhteys otettaisiin verkko-osoitteeseen, jossa tietokanta on toiminnassa (kuten MySQL). Kytkeytyminen kantaan verkko-osoitteella tarkoittaa kevyempää ylläpitoa käyttöoikeuksien osalta, koska raporttien tekijöille ei tarvitse erikseen myöntää pääsyä palvelimen hakemistoihin.

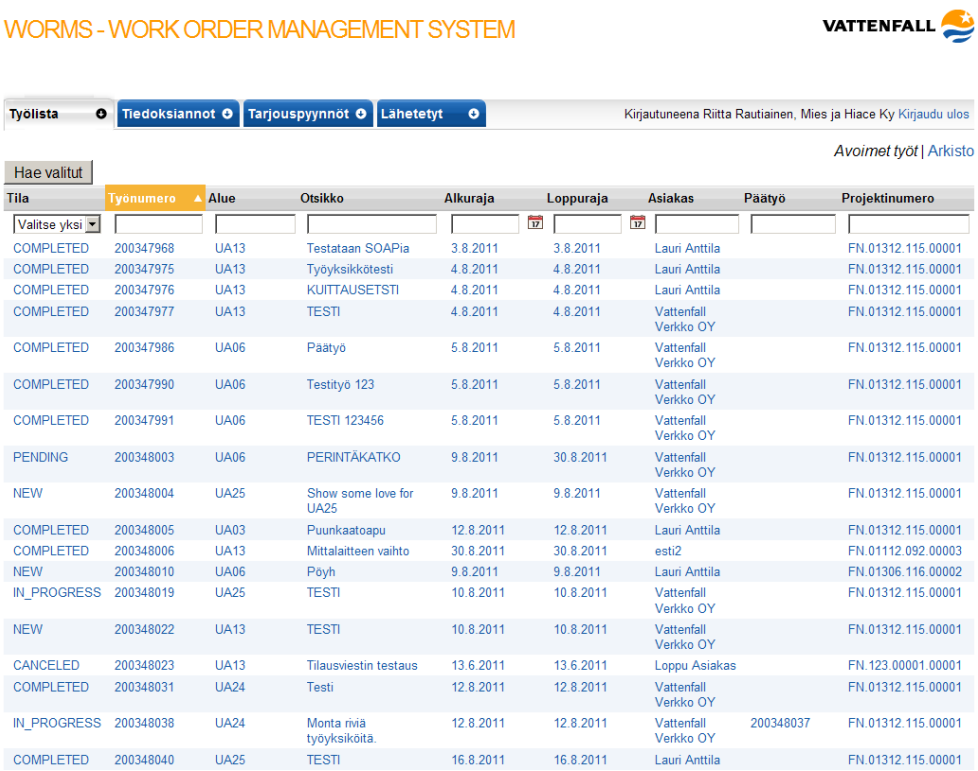
6.2 Käytettävyystestaus

Urakoitsijoilta vuosien mittaan kerätyn palautteen pohjalta Avuxin suurimpia ongelmia ovat olleet sekavuus, hitaus ja yleinen toimimattomuus. Nämä ”ominaisuudet” kattavat siis käänteisesti sen, mistä hyvä käyttäjäkokemus syntyy. Hyvä käyttäjäkokemus koostuu tarkoituksenmukaisesta toiminnallisuudesta, toimintavarmuudesta ja suorituskyvystä. Järjestelmän tulee siis tukea tehtävää työtä luomalla käyttäjälle edellytykset suorittaa haluamansa tehtävä vaivattomasti. Sen tulee olla saatavilla silloin kun sitä tarvitaan ja siirtymien sekä erilaisten transaktioiden suorittaminen pitää tapahtua nopeasti.

6.2.1 Käyttöliittymä

Testauksessa tutkittiin pienurakoitsijakäytössä olevalle Web UI:lle asettuja vaatimuksia, eli sitä voiko urakoitsija hoitaa hänelle määrätyt tehtävät käyttöliittymän puitteissa ja onko kaikki tarvittava tieto saatavilla. Esimerkki käyttöliittymänäkymästä löytyy kuvasta 14.

Käyttöliittymä on rakennettu Wormsin päälle käyttäen Apache Wicketiä, joka on valmis framework web-applikaatioiden kehittämiseen mm. Java-pohjaisille järjestelmille. Se pitää HTML-koodin erillään Java-koodista ja nämä kaksi maailmaa kommunikoivat keskenään ainoastaan Wicket-ID -tunnisteiden avulla. Wicket mahdollistaa myös sivujen helpon lokalisoinnin, joten mahdollinen suomalaisen työnohjausmallin jalostaminen Vattenfallin muihin maayhtiöihin on mahdollista pienellä vaivalla.



Tila	Työnumero	Alue	Otsikko	Alkuraja	Loppuraja	Asiakas	Päätyö	Projektinumero
Valitse yksi								
COMPLETED	200347968	UA13	Testataan SOAPia	3.8.2011	3.8.2011	Lauri Anttila		FN.01312.115.00001
COMPLETED	200347975	UA13	Työyksikkötesti	4.8.2011	4.8.2011	Lauri Anttila		FN.01312.115.00001
COMPLETED	200347976	UA13	KUITTAUSETSTI	4.8.2011	4.8.2011	Lauri Anttila		FN.01312.115.00001
COMPLETED	200347977	UA13	TESTI	4.8.2011	4.8.2011	Vattenfall Verkko OY		FN.01312.115.00001
COMPLETED	200347986	UA06	Päätyö	5.8.2011	5.8.2011	Vattenfall Verkko OY		FN.01312.115.00001
COMPLETED	200347990	UA06	Testityö 123	5.8.2011	5.8.2011	Vattenfall Verkko OY		FN.01312.115.00001
COMPLETED	200347991	UA06	TESTI 123456	5.8.2011	5.8.2011	Vattenfall Verkko OY		FN.01312.115.00001
PENDING	200348003	UA06	PERINTÄKATKO	9.8.2011	30.8.2011	Vattenfall Verkko OY		FN.01312.115.00001
NEW	200348004	UA25	Show some love for UA25	9.8.2011	9.8.2011	Vattenfall Verkko OY		FN.01312.115.00001
COMPLETED	200348005	UA03	Puunkaatoapu	12.8.2011	12.8.2011	Lauri Anttila		FN.01312.115.00001
COMPLETED	200348006	UA13	Mittalaitteen vaihto	30.8.2011	30.8.2011	esti2		FN.01112.092.00003
NEW	200348010	UA06	Pöyh	9.8.2011	9.8.2011	Lauri Anttila		FN.01306.116.00002
IN_PROGRESS	200348019	UA25	TESTI	10.8.2011	10.8.2011	Vattenfall Verkko OY		FN.01312.115.00001
NEW	200348022	UA13	TESTI	10.8.2011	10.8.2011	Vattenfall Verkko OY		FN.01312.115.00001
CANCELED	200348023	UA13	Tilausviestin testaus	13.6.2011	13.6.2011	Loppu Asiakas		FN.123.00001.00001
COMPLETED	200348031	UA24	Testi	12.8.2011	12.8.2011	Vattenfall Verkko OY		FN.01312.115.00001
IN_PROGRESS	200348038	UA24	Monta riviä työyksiköitä.	12.8.2011	12.8.2011	Vattenfall Verkko OY	200348037	FN.01312.115.00001
COMPLETED	200348040	UA25	TESTI	16.8.2011	16.8.2011	Lauri Anttila		FN.01312.115.00001

Kuva 14: Wormsin urakoitsijakäyttöliittymä, © Vattenfall Verkko 2011

Käyttöliittymä on AJAX-pohjainen, eli sivusto hakee käyttäjän tietämättä lisää sisältöä taustajärjestelmästä. Ruudulle tulee dataa sitä mukaa, kun tietoa siirtyy tietokannasta käyttöliittymään. Tämä mahdollistaa etenkin työlistan nopean avautumisen sisäänkir-

jautumisen yhteydessä. Avuxiin verrattuna muutos on merkittävä, koska Avuxin työlista avautui käyttäjälle vasta, kun kaikki tiedot oli saatu noudettua tietokannasta.

Käyttöliittymän vaatimuksia olivat:

- Uusien työtilausten, tiedoksiantojen ja tarjouspyyntöjen vastaanottaminen
- Töiden selaaminen
- Töiden ottaminen käsittelyyn
- Töiden kuittaaminen valmiiksi
- Tiedoksiantojen lukeminen ja niihin vastaaminen tiedoksiannolla
- Tarjouspyyntöjen lukeminen ja niihin vastaaminen tarjouksella
- Yllä mainittujen arkistointi

Käyttöliittymätesti suoritettiin ensin Vattenfallin toimesta, jotta selkeimmät puutteet ja ongelmat saatiin kaivettua esille. Tämän jälkeen urakoitsijan edustaja pyydettiin testaamaan käyttöliittymää omasta näkökulmastaan.

Varhaisessa testivaiheessa löytyneisiin virheisiin lukeutuivat mm. tiedon esittämiseen liittyvät puutteet. UI oli liian vaativa tilausten tietosisällön osalta, eikä esittänyt saapuneita tilauksia lainkaan työlistalla, mikäli esimerkiksi työkohteen osoitetietoihin ei ollut täytetty kaikkia mahdollisia kenttiä. Koska valtaosa tilauksista ei koskaan sisällä täydellisiä tietoja, muokattiin käyttöliittymä esittämään kaikki mahdolliset työt, mikäli vain perusehdot (työtyyppi, urakoitsijan tunniste jne) löytyivät työltä. Toisin sanoen samat tiedot vaaditaan UIssa esittämiseksi, jotka vaaditaan että SAPista saapuva sanoma ylipäätään prosessoidaan Wormsiin.

Hyväksyttävän toiminnan kriteereiksi määriteltiin seuraavat asiat:

- Kirjautumisen jälkeen työlista avautuu 2 sekunnin kuluessa
- Kaikki hallintatoiminnot (nappulat, linkit) toimivat ja ohjaavat oikeaan paikkaan
- Tehtävän valitseminen työlistalta ja avautuminen ruudulle kestää korkeintaan 2 sekuntia
- Käyttäjä ei näe kuin edustamansa yrityksen tehtävät
- Uuden tiedoksiannon lähettäminen kestää korkeintaan 2 sekuntia
- Uuden tarjouksen lähettäminen kestää korkeintaan 2 sekuntia
- Käyttäjä saa liitetiedostot auki
- Käyttäjä voi lisätä liitetiedostoja uusille tiedoksiannoille ja tarjouksille
- Käyttäjä pystyy suodattamaan työlistaa eri otsikoiden perusteella
- Käyttäjä voi arkistoida vanhoja töitä, tiedoksiantoja ja tarjouspyyntöjä

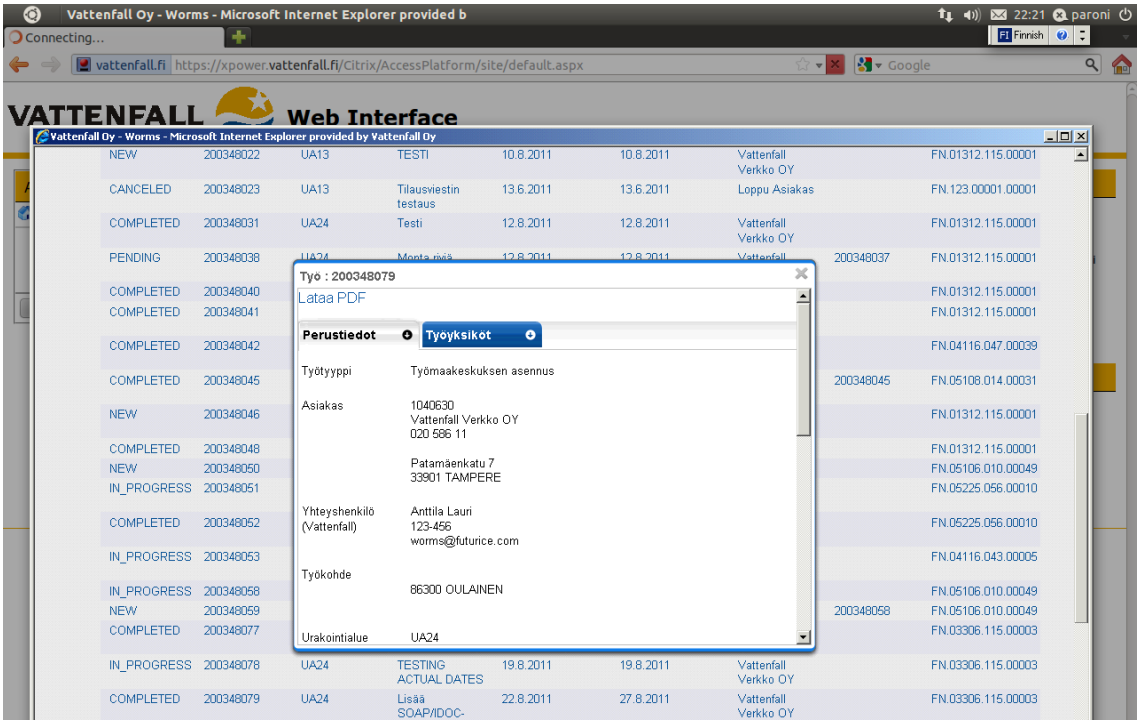
- Käyttäjä näkee Vattenfallille lähettämänsä tiedoksiannot ja tarjoukset

AJAXin avulla myös töiden etsiminen työlistalta tapahtuu tehokkaasti, koska haku on lähestulkoon reaaliaikainen eikä vaadi sivun uudelleenlatautumista. Kun käyttäjä syöttää haluamansa hakukriteerin hakukenttään, noutaa käyttöliittymä automaattisesti hakukriteeriä vastaavat työt ruudulle. Hakulausetta voi tämän jälkeen tarkentaa ja rajata tuloksia entisestään ilman, että tulokset häviäisivät hetkeksi kokonaan ruudulta.

Tarpeeksi kattavan testauksen vuoksi käyttöliittymää testattiin sekä Vattenfallin että ulkopuolisen urakoitsijan toimesta. Vaikka käyttöliittymä tulee vain urakoitsijoiden käyttöön, testattiin sitä siis kuitenkin aktiivisemmin Vattenfallin toimesta. Tämä päätös perustui paitsi kustannustehokkuuteen, myös Avuxiin, jonka osalta palaute urakoitsijakäyttöliittymän käytettävyysongelmista oli lähtöisin nimenomaan oman organisaation sisältä. Vattenfallin henkilöstöä palvelee se, että järjestelmä paitsi toimii tehokkaasti urakoitsijan näkökulmasta, sen toiminta tukee myös omaa työtä. Urakoitsijoiden kanssa paljon tekemisissä olevat henkilöt pystyivät erittäin hyvin osoittamaan mitkä osa-alueet voivat muodostua ongelmallisiksi.

Virallisesti ympäristöä ei tarvinnut testata kuin Internet Explorer 8 -selaimella, sillä kaikki urakoitsijakäyttäjät kirjautuvat järjestelmään Citrix-ympäristön kautta ja ovat sen vuoksi tasavertaisia käytössä olevan tekniikan suhteen. Citrix-palvelin määrittää käyttäjän käytössä olevan selaimen ja lisäosat, eli kaikilla on käytössää identtinen ympäristö. IE8 toimi myös järjestelmän vaatimusmäärittelyssä edellytyksenä järjestelmän toiminnalle.

Koska toimintaympäristön ja tekniikan kehittymiseen haluttiin kuitenkin varautua, testattiin järjestelmää myös muilla selaimilla, joita olivat Google Chrome, Firefox ja Opera. Lisäksi järjestelmää käytettiin myös Citrixin Linux-version kautta (kuva 15), jolloin saatiin testattua käyttöjärjestelmän vaikutukset käyttäjäkokemukseen.



Kuva 15: Wormsin käyttöliittymätestausta Ubuntu 11.04:n, Firefoxin ja Citrix Receiverin kautta avattuna.

Selainvalinta itsessään oli hieman haasteellinen ohjelmistokehityksen kannalta, koska Wicketin ominaisuuksia jouduttiin tahallaan ”huonontamaan”, jotta ne toimivat oikein IE:llä. Esimerkiksi Google Chromella kaikki ominaisuudet toimivat lähtökohtaisesti oikein, mutta IE kaatui tai palautti virheen tiettyjä sivuja avatessaan.

Käyttöliittymätestaukseen liittyy olennaisesti myös kuormatestaus, josta puhutaan luvussa 6.4.1. Käyttöliittymätesti yhdistettynä kuormatestiin antaa viitteitä siitä miten hyvin järjestelmä toimii tositilanteessa, kun sitä käyttää samanaikaisesti useita kymmeniä tai jopa satoja käyttäjiä.

6.2.2 Käyttöliittymän virheensieto

Koska voidaan olettaa käyttäjien toimivan toisinaan ohjeista poiketen, pyrittiin virheensietotestillä simuloimaan mahdollisimman monipuolisesti käyttöohjeista poikkeavia käyttötapauksia. Tavoitteena oli löytää ja korjata sellaiset tapahtumaketjut, jotka aiheuttivat järjestelmässä vääränlaista toimintaa. Vääränlaiseksi toiminnaksi määriteltiin mm. järjestelmän hidastuminen, kaatuminen ja erilaiset virheilmoitukset.

Testauksessa oli pohjimmiltaan kyse siitä, että järjestelmäteknisesti estetään kaikki mahdolliset tavat saada käyttäjälle aikaan virhetilanne, mikä vaatisi ulkopuolista apua ja siten ylimääräisiä resursseja Vattenfallin puolelta.

Samalla testattiin myös muuten vähemmälle huomiolle jätettyä tietoturvaa, koska web-käyttöliittymä on käytännössä ainoa looginen väline siirtää järjestelmään vihamielistä koodia.

Järjestelmän rakentamisen aikaan uutisissa viitattiin ”RefRef”-nimellä tunnettuun hyökkäystyökaluun, jonka hakkeriryhmä Anonymous aikoi julkaista 17. Syyskuuta 2011. Hyökkäys nojaa Apache-palvelinten haavoittuvuuteen (versiot ennen 2.2.20) ja se mahdollistaa palvelinten kaatamisen web-selaimen osoiteriviltä käsin pienellä vaivalla. Hyökkäys pyytää palvelinta lähettämään tavallisen sivun, mutta samalla vaatii että sivu toimitetaan niin pienissä osissa, että palvein kaatuu ylisuureen kuormaan. (Context Information Security Limited 2011)

Koska Wormsin palvelin sijaitsee palomuurin takana, pyörii Apache httpd:n sijaan Apache Tomcatin päällä ja sen web-käyttöliittymään on pääsy ainoastaan Citrixin kautta (ilman pääsyä osoiteriville), on järjestelmä immuuni kyseiselle hyökkäykselle.

Keväällä ja kesällä 2011 uutisoitiin myös laajalti Lulzsec-nimisen ryhmän tekemistä tietomurroista mm. CIA:n ja Sonyn tietokantoihin. Nämä murrot oli toteutettu SQL-injektioilla, eli käytetty hyväksi hyökkäyksen kohteena olleiden tahojen julkisilla palvelimilla olleita tietokantahaavoittuvuuksia. SQL-injektiossa esimerkiksi verkkosivulla

olevaan hakukenttään, jolla normaalioloissa etsitään jotain tietoa julkisista tietokannan tauluista, syötetään hakulauseen lisäksi SQL-kyselyksi muotoiltu lisälause. Mikäli järjestelmä on huonosti rakennettu, on tämän avoimen hakukentän kautta mahdollista ajaa tietokannassa minkälaisia SQL-komentoja hyvänsä.

Kuten tekstistä on aiemmin käynyt ilmi, Wormsin käyttöliittymä nojaa suorien SQL-kyselyjen sijaan niin sanottuihin RESTeihin. Käytännössä on mahdotonta ajaa minkäänlaista SQL-koodia missään Wormsin hakukentässä, koska data tulee käyttöliittymään XML-muodossa ja sitä pystytään suodattamaan ainoastaan session avaamisen yhteydessä noudettujen tietojen seasta. Myös tallennukset kantaan ajetaan RESTin läpi ja käsittelylogiikka nojaa TIEKE-sanomien käsittelyssä määriteltyihin ehtoihin, jolloin vain tiettyjen kenttien muuttaminen tietyllä tavalla on mahdollista.

Havaitut ongelmat liittyivät kuitenkin lähinnä aiemmin mainitun IE8:n toimintaan Wicketin kanssa. Nämä ongelmat ratkaistiin välittömästi.

6.3 Käyttövarmuustestaus

Avuxin suurimpia ongelma on ollut sen epävakaus ja vaikea huollettavuus. Käyttövarmuustestauksella pyrittiin saamaan järjestelmä toimimaan väärin sekä tutkimaan miten helposti virheet saa korjattua.

Wormsin tulee toimintaympäristön vuoksi pysyä toimintakuntoisena ympäri vuorokauden, vuoden jokaisena päivänä. Käyttökeskuksesta tehtäviä vikatilauksia lähetetään urakoitsijoille myös öisin ja viikonloppuisin, jolloin järjestelmätuen saaminen on hitaampaa kuin normaalina työaikana. Näin ollen on voitava luottaa siihen, että järjestelmä suorittaa sille asetetut tehtävät ilman katkoksia kaikkina vuorokauden aikoina.

Wormsin koodi pohjautuu pitkälti testattuihin ja toimiviin standardeihin, kuin myös itse tietokanta ja muut käytetyt tekniikat. Tämä luo hyvät edellytykset häiriöttömälle toiminnalle, koska kaikki osa-alueet on pyritty sovittamaan keskenään yhteensopiviksi.

Epävakautta lähdettiin hakemaan yhdistämällä mm. kohdan 6.4 suorituskykytestejä web-käyttöliittymässä samaan aikaan tehtäviin erikoisiin asioihin (sivujen jatkuva uudelleenlataus, linkkien sattumanvarainen klikkailu, yleinen huolimattomuus). Kaikista yrityksistä huolimatta järjestelmää ei ”vakio-olosuhteissa” saatu sellaiseen kuntoon, että se olisi vaatinut ylläpitotoimenpiteitä palvelimelle.

Koska todellista käyttötilannetta ei mitenkään voitu testauksen aikana simuloida, perustuu arvio hyvästä käyttövarmuudesta siihen, että testipalvelinta ei missään vaiheessa normaalitestausta jouduttu käynnistämään uudelleen, kuten ei myöskään tietokantaa tai www-palvelinta. Testit jaksottuivat kokonaisuutena kolmen kuukauden ajanjaksolle ja palvelimelle syötettiin niin paljon virheellisiä ja oikeita sanomia, että niiden olisi pitänyt aiheuttaa jotain ongelmia, mikäli järjestelmän vikasietoisuus olisi huono.

Jotta testeistä olisi saatu edes jotain negatiivisia tuloksia, tehtiin sille ennen käyttöönottoa niin massiivinen http-kuorma, että palvelin saatiin lopulta kaatumaan. Tilanne simuloi jo palvelunestohyökkäystä, jonka kohdistuminen suoraan Wormsiin ei käytännössä ole mahdollista. Lisäksi vaikka Citrix-ympäristö saataisiin talon ulkopuolelta kaadettua, säilyy kriittinen TIEKE-rajapintaa hyödyntävä liikenne toimintakykyisenä ja vikatilaukset saadaan välitettyä isoimmille toimijoille.

Käyttövarmuutta tullaan seuraamaan tarkasti järjestelmän käyttöönoton jälkeen ja arvioidaan mahdollisesti ilmenneitä ongelmia yhdessä ohjelmistotoimittajan kanssa.

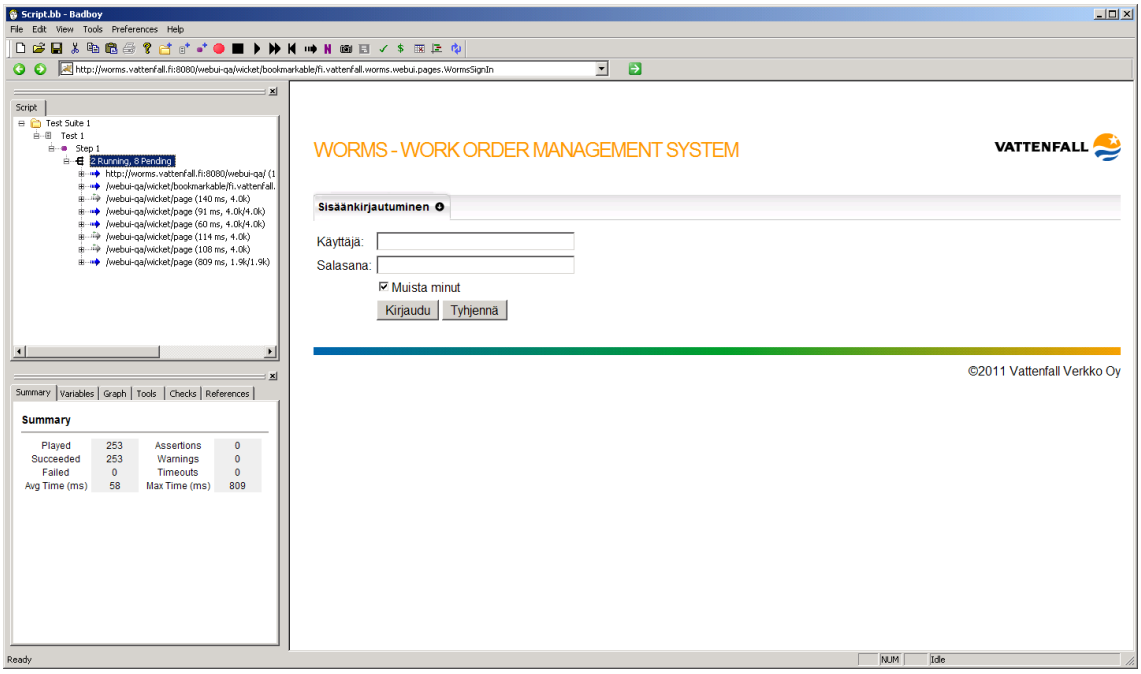
6.4 Suorituskykytestaus

Suorituskykytestauksessa järjestelmälle pyrittiin simuloimaan mahdollisimman suurta kuormaa ja selvittämään palvelimen toimintakyvyn rajat. Avuxin ongelmana on ollut suurten tietomäärien käsittely, joten kuormatestauksella haluttiin varmistua Wormsin kyvystä lähettää ja vastaanottaa samanaikaisesti jopa epätodellisen suuria määriä viestejä. Suurten myrskyjen aikaan tilauksia voi lähteä jopa tuhat kappaletta vuorokaudessa normaalin kuorman lisäksi.

6.4.1 HTTP-kuormatesti

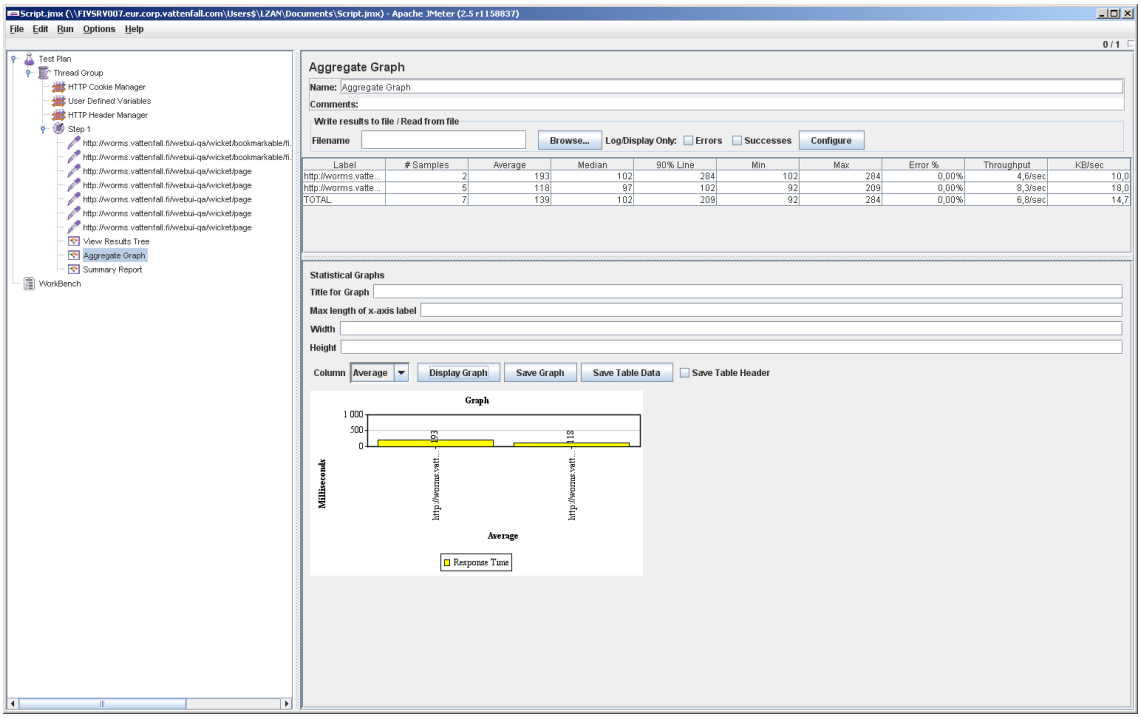
Yksi Avuxin korvaamisen parhaita puolia oli ulkoisten käyttäjien määrän radikaali vähentäminen. Isoimpien toimijoiden kaikkien töiden siirtäminen sähköiseen rajapintaan vähentää tarvetta ulkoisille käyttäjätunnuksille (ja käyttäjille) n. 700 tunnuksen verran. Web-käyttöliittymää käyttää vain pieni joukko urakoitsijoita, joilla on myös isompia kilpailijoitaan pienempi työkuorma. Näin ollen web-palvelun tarve käsitellä suuria määriä käyttäjiä oli lähtökohtaisesti vaatimaton. Jotta järjestelmästä saatiin kuitenkin järkevä testidata ulos, testattiin palvelinta siten että sille löydettiin raja-arvot liikenteelle jolloin työskentely on vielä tehokasta (ns ”kriittinen suorituskyky”) ja se milloin järjestelmä kaatuu kokonaan.

Kuormaa piti alun perin simuloida Apache JMeter-ohjelmistolla, joka on yleisesti käytetty avoimen lähdekoodin testiohjelmisto. JMeter ei kuitenkaan tue AJAX-pohjaisten sivustojen simulointia tarpeeksi hyvin, joten sille jouduttiin etsimään vaihtoehto. Vaihtoehtoiseksi työkaluksi valikoitui JMeterin tapaan ilmainen Badboy, jolla voidaan nauhoittaa ohjelmiston oman selaimen sisällä tapahtuva toiminta ja rakentaa siitä testimalli. Kuvassa 16 on esimerkki Badboyn käyttöliittymästä.



Kuva 16: Badboyn käyttöliittymä, jossa automatisoitu käyttötapaustestaus käynnissä

Badboy itsessään ei sovellu raskaaseen kuormatestaukseen. Sillä voidaan kyllä simuloida hyvin käyttötapauksia, mutta massiivista kuormaa ja siihen liittyvää raportointia sillä ei pysty hallitsemaan tehokkaasti. Ohjelmisto omasi kuitenkin vakiona tuen testitapausten tallentamiseksi JMeterin ymmärtämään muotoon. Tämä mahdollisti todellisen käyttötapauksen massasimuloinnin myös alkuperäisen suunnitelman mukaan. Kuvassa 17 on esitettynä kuvan 16 testitapauksesta muodostettu JMeter-testi, johon on lisäksi kytketty päälle useita ”monitoreja”.



Kuva 17: Apache JMeterin käyttöliittymä, jossa testitapauksesta muodostunutta lokitietoa.

Yksi testitapaus oli esimerkiksi tilanne, jossa kymmeniä käyttäjiä kirjautuu yhtaikaisesti Wormsiin, klikkailee useita linkkejä, vaihtaa työlistan näkymiä sekä kirjautuu sitten ulos järjestelmästä.

Paitsi että testaus pyöri koneellisesti, testattiin järjestelmää samaan aikaan myös manuaalisesti muutaman käyttäjän toimesta, jotta voitiin havaita miten tietynlainen kuorma vaikuttaa todellisuudessa käyttäjän kokemaan toimintaan.

Testeillä ei varsinaisesti saatu aikaan tilannetta, että järjestelmä olisi ollut millään muotoa hidas tai epävakaa. Ainoastaan mentäessä teoreettisiin maksimikuormiin (tuhansia käyttäjiä), alkoi palvelin hylätä pyyntöjä. Tämä ei kuitenkaan näkynyt palvelimen muistinkäsittelyssä, joka on ollut Avuxin ongelma. Taustajärjestelmä sinänsä toimii hyvin, ainoastaan käyttöliittymä ei aina pääse käsiksi tietoihin.

6.4.2 SOAP-kuormatesti

Jokainen muutos työtilauksille joko SAPissa tai Wormsissa (UI:n tai XML-rajapinnan kautta) aiheuttaa SOAP-liikennettä järjestelmien välille. Laskennallisesti on odotettavissa, että viestejä liikkuu toimistoaikoina keskimäärin n. 9 kpl minuutissa. Tämä saadaan arvioimalla teoreettinen maksimikuorma normaaliolosuhteissa, eli 1000 tilaussanomaa päivässä, joille jokaiselle tehdään neljä muutosta saman päivän aikana. Näin saadaan 4000 päivitystä vuorokaudessa, joka jaettuna varsinaiselle toimistoajalle on 8,3 viestiä minuutissa.

Näin pieni määrä ei varsinaisesti ole mikään ongelma tietokannalle tai Camelin käsittelylogiikalle, mutta entäpä jos viestejä tulee yhdellä kertaa todella paljon?

Testi suoritettiin generoimalla ISUun koneellisesti 100 kpl työtilauksia, jotka kaikki vapautettiin samanaikaisesti. Tämän seurauksena syntyi 100 IDOC/SOAP -sanomaa, jotka siirtyivät Wormsiin. JMeteriä ei käytetty, koska sen SOAP-prosessointi saadaan toimimaan vain staattisella testidatalla, jolloin käytännössä 30 käyttäjää tekisi samoja muutoksia samalle tilaukselle saman aikaan, joka ei teknisesti ole mahdollista SAPissa. JMeter soveltuukin SOAP-toiminnallisuuksiltaan paremmin sellaisen web-palvelun testaamiseen, jossa käyttäjät todellisuudessa voivat tehdä samantyyppisiä kyselyitä samaan aikaan.

Viestien saapuminen todennettiin Wiresharkilla, jota käytettiin myös mahdollisten ongelmien havaitsemiseksi jo vastaanottovaiheessa. Lisäksi Wiresharkilla pystyttiin tar-

kasti mittaamaan kuinka pitkä aika ensimmäisen ja viimeisen saapuneen viestin välillä kului, eli kauanko XI prosessoi isoa määrää IDOC->SOAP muunnoksia.

Kun viestit oli saatu prosessoitua ilman virheitä, tarkistettiin vielä niiden tila Wormsin tietokannassa. Onnistumisen kriteerinä oli saada tietokantaan sata uutta tilausta, jotka ovat identtisiä kaikilta muilta osin, paitsi muuttuvan sapID:n osalta. Vaikeusasteen nostamiseksi tilaukset oli kohdistettu myös TIEKE-sanomia hyödyntävälle urakoitsijalle, jolloin voitiin myös nähdä miten Camel suoriutui viestimuunnoksista näin suuren volyymin osalta.

SAPin XI-integraatiopalvelin käsittelee IDOC->SOAP -muunnoksia sen verran hyvin, että Worms ei missään vaiheessa tukehtunut saapuvien viestien määrään. XI muodostaa viesteille selkeän käsittelyjonon, eikä yritä lähettää kaikkea tietoa kerralla. Testi oli kaikin puolin onnistunut.

6.4.3 XML-kuormatesti

Tällä testillä simuloitiin tilannetta, jossa FTP-palvelimelle ilmestyy kerralla iso määrä urakoitsijoiden järjestelmistä lähteneitä TIEKE XML-sanomia. Tämä skenaario on mahdollinen, koska välityspalveluntarjoaja välittää viestit nipuissa ja on mahdollista että viestejä saapuu normaaliolosuhteissa kerralla useita kymmeniä.

Testin tarkoituksena oli selvittää miten:

- Worms yleisesti selviytyy suuresta määrästä viestejä
- tallentuuko viestien data oikein tietokantaan
- muuttuuko tilausten status oikeaksi
- kulkevatko muutokset normaalisti SAPin työtilauksille SOAP-viesteinä
- hidastaako suuri viestimäärä web-käyttäjien toimintaa

Testi suoritettiin pudottamalla suuri määrä erityyppisiä TIEKE-sanomia FTP-palvelimen saapuvien viestien hakemistoon, josta Worms prosessoi ne sisään Apache Camelin päälle rakennetun tilakoneen avulla. Testi mittasi Camelin suorituskykyä ja järjestelmän yleiselle toiminalle mahdollisesti aiheutunutta haittaa. Suorituskykyä tarkkailtiin

siten, että yksi käyttäjä toimi web-käyttöliittymässä ja teki töitä ”normaaliin tapaan”, toinen käyttäjä seurasi tilauksille tapahtuneita muutoksia SAP-järjestelmässä ja kolmas seurasi ainoastaan tietokantaan tehtyjä muutoksia sekä tarkkaili FTP-hakemiston käyttäytymistä.

Web-käyttäjä arvioi järjestelmän toimintaa ennen pudotusta, sen aikana, sekä hieman sen jälkeen. Tuloksia verrattiin normaaliin käytettävyystestauksessa mitattuun toimintaan. SAP-käyttäjä tarkkaili miten nopeasti TIEKE-sanomien sisältämä tieto saavutti SAPin työtilaukset, eli kuinka nopeasti SOAP-viestit generoitiin Wormsin toimesta.

Tietokantaa ja FTP:tä tarkkailleen käyttäjän vastuulla oli valvoa että viestit luettiin sisään FTP:ltä ja että oikeanlaiset muutokset tallentuivat oikeille tilauksille tietokannassa. Lisäksi tarkistettiin, että töiden status muuttui tilakoneen määritysten mukaiseksi.

Varmuuden vuoksi testattiin myös Wormsin kykyä suoriutua suurista liitetiedostoista, etenkin jos niitä on monen sanoman sisällä. Urakoitsijoille annettavassa sähköisen rajapinnan soveltamisohjeessa liitetiedostojen maksimikoko määritellään, mutta voi olla tilanteita jolloin urakoitsijan suunnasta lähetetään vahingossa todella isoja tiedostoja. Yleisimpiä ovat bittikarttamuodossa olevat kuvakaappaukset, jotka ovat helposti n. 10MB/kuva.

Kuten XI:n SOAP-prosessointi, myös Wormsin viestikäsittely perustuu jonoihin. Testiviestit luetaan FTP:ltä sisään yksi kerrallaan, jolloin ruuhkaa ei pääse syntymään. Mikäli jossain sanomassa on virheitä, siirtyy ko. viesti virhekäsittelykansioon ja muiden viestin käsittely jatkuu normaaliin tapaan. Käyttäjälle ei aiheutunut prosessoinnista ongelmia.

6.5 Huollettavuus ja jatkokehitys

FURPS-mallin ”viimeisenä vaiheena” Wormsia analysoidaan sen tulevaisuuden ylläpidon, huoltotoimien ja jatkokehityksen osalta. Järjestelmä rakennettiin alusta alkaen

siten, että siinä on mahdollisimman vähän kovakoodattuja, mahdollisesti tulevaisuudessa muuttuvia arvoja, sekä vältettiin liian tiukkoja integrointeja muihin järjestelmiin.

Käytännössä Worms voidaan siirtää täysin toiseen toimintaympäristöön ja korvata SAP toisella toiminnanohjausjärjestelmällä, mutta Wormsiin ei tarvita muita muutoksia kuin SOAP-rajapinnan liittymämääritysten päivitys.

Järjestelmä pohjautuu Java-koodiin, joka on yksi yleisimmin tunnetuista ohjelmointikielistä. Kommentoitu, oliopohjainen koodaus mahdollistaa tulevaisuudessa järjestelmää aiemmin tuntemattomien henkilöiden nopean osallistumisen kehitystyöhön, jolloin Vattenfall ei ole riippuvainen alkuperäisestä ohjelmistotoimittajasta. Integraatiot eri järjestelmiin on pidetty mahdollisimman yksinkertaisina. Esimerkiksi Avuxin ja SAPin välillä oli yhteensä seitsemän XML-RPC-liittymää, mutta Wormsissa SOAP-liittymiä on vain kolme.

Järjestelmä pohjautuu pitkälti avoimen lähdekoodin tunnettuihin ratkaisuihin, kuten Apache Camel, Apache Tomcat ja MySQL. Tämä asettaa järjestelmän ylläpitäjille vaatimuksia osaamiselle, mutta samalla minimoi lisenssikulut. Tietokanta on yleisimpiä käytössä olevia ratkaisuja ja siihen päästään käsiksi käytännössä katsoen kaikilla markkinoilla olevilla raporttityökaluilla.

Ainoat varsinaiset järjestelmäpäivitykset liittyvät Windows Serverin päivityksiin, joita asennetaan ennalta määrättyjen huoltoikkunoiden aikana. Näin minimoidaan käyttäjille aiheutuvia haittoja. Käytännössä päivitykset voidaan ajoittaa samaan aikaan SAPin huoltoikkunoiden kanssa, jolloin työnohjauksen käyttäminen olisi muutenkin mahdotonta.

Järjestelmästä on myös oma testipalvelin, jolla tulevat päivitykset voidaan testata ennen tuotantoon vientiä. Wormsin arkkitehtuuri mahdollistaa viimeisimmän toimivan version käyttöönoton todella nopeasti ja testijärjestelmä toimii myös tuotantopalvelimen varakoneena. Laiterikon sattuessa uusi laite saadaan käyttöön nopeimmillaan

puolessa tunnissa, kunhan tietokannan viimeisin varmuuskopio saadaan kopioitua testikoneelle ja tuotantoversion build ladattua testiversion tilalle Tomcatissa.

Vattenfallin IT-tuelle laadittiin erillinen ylläpitäjän manuaali, johon kirjattiin tunnistetut virhetilanteet joita järjestelmässä voi käyttöönoton jälkeen syntyä sekä tapauskohtaiset menettelymallit esimerkiksi missä tilanteessa Tomcatilla pyörivät applikaatiot on syytä käynnistää uudestaan tai kuinka SOAP-yhteydet saadaan pystyyn liittymäkatkoksen jälkeen.

Järjestelmästä laadittiin myös toiminnallisuuskuvaukset täydennettynä UML-kaavioina, joista selviää miten järjestelmä todellisuudessa toimii, mitä liittymiä sillä on, sekä mihin tietokannan tauluihin ja kenttiin data tallennetaan. Näin ollen, mikäli järjestelmää kehitetään jonkun muun kuin projektissa mukana olleiden tahojen toimesta, saadaan nopeasti selvitettyä kokonaiskuva järjestelmän toiminnasta tukemaan ohjelmointityötä.

Jatkokehitystä varten laaditaan erillinen toimenpideohjeistus, jonka tarkoituksena on jakaa kehitys esimerkiksi ”kvartaaliversioiksi”, jolloin uudet ominaisuudet kerätään vaatimuslistalle ja ne toteutetaan kerralla, yhtenä kehityssprinttinä ja julkaistaan esim. kolmen kuukauden välein. Näin ollen päästään irti siitä, että yksittäisiä ominaisuuksia lisätään aina kun tarve ilmenee, jolloin kokonaisuuden hallinta muuttuu vaikeaksi. Useamman ominaisuuden rakentaminen kerralla takaa paremman yhteispelin ko. ominaisuuksien ja jo toiminnassa olevan järjestelmän välillä.

7 JOHTOPÄÄTÖKSET JA POHDINTA

Ohjelmistojen testaaminen FUPRS+ -mallilla käsittää todella laajan skaalan erilaisia asioita ja tehtäviä, joiden tunnollinen suorittaminen tuottaa merkittävästi paremman lopputuloksen, kuin pintapuolinen, tärkeimmät osa-alueet kattava perustason analyysi. Malli on erittäin looginen tapa tarkastella tuotteen laatua, mutta jostain syystä se tuntuu monella ohjelmistotalolla unohtuneen.

Markkinoilla toimii jo useita laadunvarmistukseen keskittyviä palveluyrityksiä, mm. suomalainen Quentinel, jotka tarjoavat ohjelmistotestauspalvelua asiakkailleen. Kuitenkin ulkopuolisen toimijan käyttö pienen mittakaavan IT-projektissa voi mutkistaa muuten ketterän toiminnan ja aiheuttaa mittavia kustannuksia asiakasyrityksille. Kuten työn alkupäässä esitellystä VR:n tapauksesta käy ilmi, testauksen ulkoistaminen ei aina edes tuota laadukasta lopputulosta. Oma organisaatio on yleensä paras testaamaan tuotetta, koska sillä on paras tieto siitä miten sen tulisi toimia.

Testisuunnitelmien laatiminen ketterälle ohjelmistoprojektille on oma lukunsa. Aikataulutettuja testejä on käytännössä mahdotonta tehdä ennen lopullisen version valmistumista, koska eri osa-alueita testataan hyvin tehokkaasti ristiin projektin kaikissa vaiheissa. Lisäksi monissa alan kirjoissa ja käytännön kautta olen törmännyt siihen, että testaamisesta tehdään mahdollisimman monimutkaista mm. testausraporttien muodossa. Usein näiden testityökalujen käyttö on testaajalle niin vaikeaa, että valtaosa testausajasta kuluu työkalun kanssa tuskailuun, kun pitäisi keskittyä olennaiseen.

Wormsin kaltaisen – pienen ja ketterän – sovelluksen testaamisessa ei ollut mitään mieltä lähteä rakentamaan kohta kohdalta läpikäytävää testilistaa yksittäisistä käyttötapauksista ja raportoida löydöksiä sen pohjalta täyttämällä isoja kuponkeja. Tämä perustuu omaan kokemukseeni ERP-järjestelmän käyttöönottoprojektista, jossa ongelmien raportointiin ja itse raportoinnin opetteluun kulutettiin enemmän aikaa, kuin itse testaamiseen. Koska resurssit olivat rajalliset, tuli kaikki käytettävissä oleva aika keskittää itse testaamiseen. Käytännössä kaikki testaus pohjautui määrittelydokument-

tiin, löydökset kerättiin teemoittain ranskalaisiksi viivoiksi ja toimitettiin ohjelmistotoimittajan projektipäällikölle. Ketterä malli toimi hyvin, joskin tämän tyyppinen jatkuva testaus tarkoittaa monien toiminnallisuuksien testaamista aina uudestaan, vaikka ne toimisivatkin lähtökohtaisesti oikein. Muihin ohjelmistokomponentteihin tehtävät muutokset saattavat vaikuttaa yllättävästi jo aiemmin hyvin toimineisiin osiin.

Rajanveto toiminnallisuus- ja käytettävyystestaukselle Wormsin www-käyttöliittymän osalta oli hivenen hankalaa. Teknisesti kaikki tapahtuu Wormsin taustalla, eli toiminnallisuus painottuu ohjelmistokoodiin ja toimintasääntöihin. Näin ollen testit kategorisoitiin pääasiassa toiminnallisuustestauksen sisälle. Kuitenkin käyttöliittymän osalta testattiin toiminnallisuuksia, joilla luotiin uusia arvoja Wormsin tietokantaan perustuen käyttäjän syöttämiin arvoihin. Näin ollen käytettävyystestauksen rinnalla tehtiin osittaista toiminnallisuustestausta, kun varmistettiin että taustajärjestelmä toimii kuten pitää.

Opinnäytetyön tekemisen jälkeen olen hyvin vakuuttunut siitä, että yritykset voivat hoitaa tuotteiden testauksen omatoimisesti, mikäli henkilöstöstä löytyy tarpeeksi teknistä asiantuntemusta ja halua perehtyä testausfilosofiaan hieman pintaa syvemmältä.

Olennaista omatoimisesti suoritetussa testauksessa on tehdä riittävä määrä testejä ja huomioida potentiaaliset erikoistilanteet, vaikkeivät ne olisi todennäköisiä sen hetkisessä toimintaympäristössä. Muutokset liiketoiminnassa ja taustajärjestelmissä aiheuttavat nopeasti tarpeita muuttaa esimerkiksi SOAP-viestien sisältöä, jolloin mahdolliset erikoismerkit voivat sotkea koko toiminnan.

Yhteenvetona voisi todeta, että näinkin yksinkertaisen järjestelmän toiminnan testaaminen vaatii hyvää suunnittelua, vie paljon aikaa ja muistettavia asioita on paljon. Testejä ei pidä rajoittaa kattamaan ainoastaan yleisimmät ja odotettavissa olevat käyttö- ja virhetapaukset, vaan huomiota tulee kiinnittää myös ”mahdottomaan”. Erilaisten syöttövirheiden ja järjestelmien osien kaatuminen voi aiheuttaa sekä SAPissa että Wormsissa yllättäviä tilanteita, jotka pystytään testauksen jälkeen hallitsemaan huomattavasti paremmin.

Työ ei suinkaan lopu tämän opinnäytetyön valmistumiseen. Järjestelmän käyttöönotto on jännittävä tilanne ja vasta viikkojen tai kuukausien jälkeen voidaan olla varmoja siitä, että järjestelmä todella toimii oikein. Kun mukaan otetaan liki kymmenen ulkoista kumppania, voi ulkopuolisista järjestelmistä saapuva virheellinen data tuottaakin sellaisia tapahtumia järjestelmässä, joita ei ole osattu ennakoida.

Tietokannan koon kasvaessa myös tiedon käsittely on asia, johon pitää kiinnittää erityishuomiota. Toimivatko kaikki käsittelysäännöt oikein, kun dataa on kannassa vuoden tai kahden edestä? Testien perusteella voidaan olla luottavaisin mielin, että näin on, mutta todellisuus ei aina kohtaa testien kanssa.

Laadunvarmistus jatkuu siis Wormsin elinkaaren loppuun asti. Järjestelmän normaalin seurannan lisäksi lisäominaisuuksien suunnitteluun tulee kiinnittää erityistä huomiota, etteivät muutokset sotke hyvin toimivan järjestelmän rakennetta. Varmuuskopioinnista ja järjestelmän palautuksesta erilaisten laiterikkojen vuoksi on huolehdittava. Kun Worms aikanaan tulee elinkaarensa päähän, pitää myös datan konversiolle uuteen järjestelmään olla suunnitelma valmiina. Tätä päivää silmällä pitäen tietokantarakenne onkin syytä pitää selkeänä. Kaikkea toimintaa tukee ajan tasalla oleva järjestelmäkuvaus, jonka avulla nykyiset ja tulevat kehittäjät ja ylläpitäjät pystyvät ylläpitämään osaamistaan ja tietoutta järjestelmän historiasta.

LÄHTEET

Agile Manifesto. 2001. http://www.agilemanifesto.org. Luettu 20.1.2011

Context Information Security Limited. 2011. Threat Intelligence Advisory – Day of Rage 17/09/11. UK

Eeles, Peter. 2005. Capturing Architectural Requirements. http://www.ibm.com/developerworks/rational/library/4706.html. Luettu 28.2.2011

Energiamarkkinavirasto. Yleistä sähkömarkkinoista. http://www.energiamarkkinavirasto.fi/select.asp?gid=38. Luettu 19.1.2011.

Gerush, Mary. 2010. Best Practices: Your Ten-Step Program To Improve Requirements And Deliver Better Software. Forrester Research: USA

Hannula, Esko. 2010. Ketteryys kohtaa todellisuuden. http://www.cs.tut.fi/tapahtumat/testaus10/Hannula.pdf. Luettu 25.1.2011

Hass, Anne Mette Jonassen. 2008. Guide to Advanced Software testing. Artech House: USA.

Holub, Allen. 2003. Why Getter And Setter Methods Are Evil. Java World. http://www.javaworld.com/javaworld/jw-09-2003/jw-0905-toolbox.html. Luettu 14.6.2011.

Kalliosaari, Kati. 2011. VR aliarvioi pahasti lippujensa kysynnän. Aamulehti 20.9.2011. Suomi

Leffingwell, Dean. 2010. Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley Professional: USA

Lekman, Lare. 2011. Lisää onnistumisia Scrumilla. Projektitoiminta 1/2011. Projektiyhdistys ry: Suomi

Mext Oy. 2011. Ohjelmistoprojektit 2010 -tutkimus. Suomi.

Pham, Andrew. 2011. Scrum in Action: Agile Software Project Management and Development. Course Technology PTR: USA

Schwaber, Ken & Sutherland, Jeff. 2010. Scrum Guide. Scrum.org

TIEKE ry. 2010. Energiateollisuuden tiedonsiirtosuositus 1.0. Suomi.

Valtiontalouden tarkastusvirasto. 2011. Valtiontalouden tarkastusviraston tuloksellisuustarkastuskertomus 217/2011: Sosiaali- ja terveydenhuollon valtakunnallisten IT-hankkeiden toteuttaminen.

Vattenfall. Vattenfallin historia Suomessa. http://www.vattefall.fi. Luettu 19.1.2011.

Vainio, Riitta. 2011. Moni joutui liputta junaan. Helsingin Sanomat 20.9.2011. Suomi

Wikipedia. 2011. Planning Poker. http://en.wikipedia.org/wiki/Planning_poker Luettu 8.6.2011

LIITTEET

LIITE 1: Avuxin käyttäjäkysely

Avuxin käyttäjäkysely

Kysely suljetaan 10.12.2010

Ikääntyneestä Avux-työnhallinnasta tullaan luopumaan vuoden 2011 aikana. Korvaavana järjestelmänä tullaan käyttämään SAPin PM-moduulia (tuttavallisemmin IS-Un työtilauksia) sekä mahdollisesti hyödyntää SAP KIS:n tuomia ominaisuuksia.

Projektin tavoitteena on

- Yksinkertaistaa työnohjausta vähentämällä järjestelmiä
- Parantaa toimintavarmuutta
- Ohjata kaikki työt suoraan urakoitsijoiden järjestelmiin
- Parantaa raportointia ja töiden seurantaa

Jotta muutos saadaan hoidettua mahdollisimman sujuvasti, kerätään nyt Avuxin käyttäjiltä palautetta ja ehdotuksia työnhallinnan kehittämiseksi.

Vastaaminen vie noin 5-10 minuuttia.

Kyselyssä on 14 kysymystä.

Peruskysymykset

1 [0001]Missä työskentelet? *

Valitse **vain yksi** seuraavista:

- ○ Yksityisasiakkaat
- ○ Sähkönmyyjänvaihdot
- ○ Energiatiedot
- ○ Yritysasiakkaat
- ○ Asiakastilaukset
- ○ Tekninen asiakaspalvelu
- ○ Käyttökeskus
- ○ Käytön suunnittelu
- ○ Strateginen suunnittelu
- ○ Sähköinen suunnittelu
- ○ Rakennuttaminen ja materiaalit
- ○ Sähköasemat ja voimajohdot
- ○ Maankäyttö- ja verkkotiedot
- ○ Kenttä (kaikki alueet)
- ○ Muu [ ]

2 [0002]Arvioi kuinka suuren osan (%) työnohjaukseen liittyvistä tehtävistä teet tällä hetkellä Avuxissa *

Vastauksesi:

3 [0003]Käytätkö Avuxin vakioraportteja *

Valitse **vain yksi** seuraavista:

- ○ Kyllä
- ○ Ei

4 [0004]Käytätkö muita työnohjaukseen liittyviä raportteja (esim. SAPista)? *

Valitse **vain yksi** seuraavista:

- ○ Kyllä
- ○ Ei

Töiden hallinta

5 [0001]Valitse viisi sinulle tärkeintä ominaisuutta työnhallinnassa

Valitse **korkeintaan** 5 vastausta:

- [ ] Nopeus
- [ ] Toimintavarmuus
- [ ] Selkeys
- [ ] Työyksiköiden käsittely
- [ ] Helppokäyttöisyys
- [ ] Tietojen oikeellisuus
- [ ] Automaattiset töiden perustumiset (esim. rakentamisen viimeistely)
- [ ] Työtyyppikohtaiset näkymät
- [ ] Liitetiedostojen käsittely
- [ ] Yksinkertainen laskuntarkastus
- [ ] Tekstiviestitoiminnot
- [ ] Mobiilikäyttö (töiden kuittaus kännykällä)
- [ ] Sähköposti-ilmoitukset töille tehdyistä muutoksista
- [ ] Raportit
- [ ] Muu: _____

6 [0002]Edelliseen kysymykseen liittyen, valitse viisi vähiten tärkeää ominaisuutta

Valitse **korkeintaan** 5 vastausta:

- [ ] Nopeus
- [ ] Toimintavarmuus
- [ ] Selkeys
- [ ] Työyksiköiden käsittely
- [ ] Helppokäyttöisyys
- [ ] Tietojen oikeellisuus
- [ ] Automaattiset töiden perustumiset (esim. rakentamisen viimeistely)
- [ ] Työtyyppikohtaiset näkymät
- [ ] Liitetiedostojen käsittely
- [ ] Yksinkertainen laskuntarkastus
- [ ] Tekstiviestitoiminnot
- [ ] Mobiilikäyttö (töiden kuittaus kännykällä)
- [ ] Sähköposti-ilmoitukset töille tehdyistä muutoksista

- [ ] Raportit
- [ ] Muu:

Raportit

7 [0001]

Pisteytä seuraavat raportteihin liittyvät ominaisuudet (1 ei tärkeä, 5 erittäin tärkeä)

Valitse sopivin vaihtoehto:

	1	2	3	4	5
Raporttien selkeys	○	○	○	○	○
Raportin standardimuoto (Excel, PDF jne.)	○	○	○	○	○
Raporttien vastaanotto sähköpostitse	○	○	○	○	○
Raporttien tarkastelu internetin kautta	○	○	○	○	○
Graafiset ominaisuudet (esim. pylväs- tai piirakkakaaviot)	○	○	○	○	○
Raporttien helppo muunneltavuus (esim. uudet kentät)	○	○	○	○	○

8 [0002]Minkälaisia raporttitarpeita sinulla on? Esim. kustannusseuranta, töiden aikataulujen pitävyys jne.

Vastauksesi:

Urakoitsijakäyttö

9 [0001]Arvioi seuraavat urakoitisjoita koskevat väittämät (1 täysin eri mieltä, 5 täysin samaa mieltä)

Valitse sopivin vaihtoehto:

	1	2	3	4	5
Työtilausten välittäminen urakoitsijoille tulee olla nopeaa ja varmaa	○	○	○	○	○
Urakoitsijoiden tulisi toimia ensisijaisesti omassa järjestelmässään	○	○	○	○	○
VFV:n tulee tarjota urakoitsijalle monipuolinen työnohjausportaali (vrt. Avux)	○	○	○	○	○
Urakoitsijan edustajan (esim. työnjohtaja) tehtävä on huolehtia töiden välittämisestä työntekijöille	○	○	○	○	○

Vapaa sana

10 [0001]
Mitä työnhallinnan kannalta olennaista Avuxista on puuttunut? Kerro vapaasti mitä tarpeita sinulla on.

Vastauksesi:

11 [0002]Mitä Avuxin toimintoja pidät turhina? Kerro vapaasti mitä voitaisiin karsia.

Vastauksesi:

12 [0003]Avux on ollut käytössä pitkään. Kerro vapaasti mitä huolia liittyy sen korvaamiseen muilla järjestelmillä.

Vastauksesi:

Projektin eteneminen

Kyseessä on joiltakin osin iso muutos nykyiseen työnhallintaan. Projektin onnistumisen kannalta on olennaista, että käyttäjien mielipiteet ja asiantuntemus huomioidaan, jotta lopputuloksena olisi toimiva järjestelmä.

13 [0001]Haluan seurata projektin etenemistä ja tilaan uutiskirjeen sähköpostiini (täytä osoite)

Vastauksesi:

14 [0002]Olen kiinnostunut osallistumaan myös kehitykseen ja testaukseen *

Valitse **vain yksi** seuraavista:

- ○ Kyllä
- ○ Ei

Kiitos vastauksestasi!
01.01.1970 – 02:00

Lähetä vastaukset.
Kiitos vastauksistasi!.